Backpropagation and Neural Nets

EECS 442 – Jeong Joon Park
Winter 2024, University of Michigan

So Far: Linear Models

$$L(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

- Example: find **w** minimizing squared error over data
  - Each datapoint represented by some vector **x**
    - Can find optimal **w** with ~10 line derivation

Last Class

$$L(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}; \boldsymbol{x}))$$

- What about an arbitrary loss function L?
- What about an arbitrary parametric function f?
- Solution: take the gradient, do gradient descent

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_{\boldsymbol{i}} - \alpha \nabla_w \underbrace{L(f(\boldsymbol{w}_i))}$$

What if L(f(w)) is complicated?

Today!

Taking the Gradient – Review

$$f(x) = (-x + 3)^2$$

$$f = q^2 \qquad q = r + 3 \qquad r = -x$$

$$\frac{\partial f}{\partial q} = 2q \qquad \frac{\partial q}{\partial r} = 1 \qquad \frac{\partial r}{\partial x} = -1$$

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial r}\frac{\partial r}{\partial x}} = 2q * 1 * -1$$

Chain rule

$$= -2(-x + 3)$$

$$= 2x - 6$$

Supplemental Reading

- Lectures can only introduce you to a topic
- You will solidify your knowledge by **doing**
- I highly recommend working through everything in the Stanford CS213N resources
  - http://cs231n.github.io/optimization-2/
- These slides follow the general examples with a few modifications. The primary difference is that I define local variables n, m per-block.

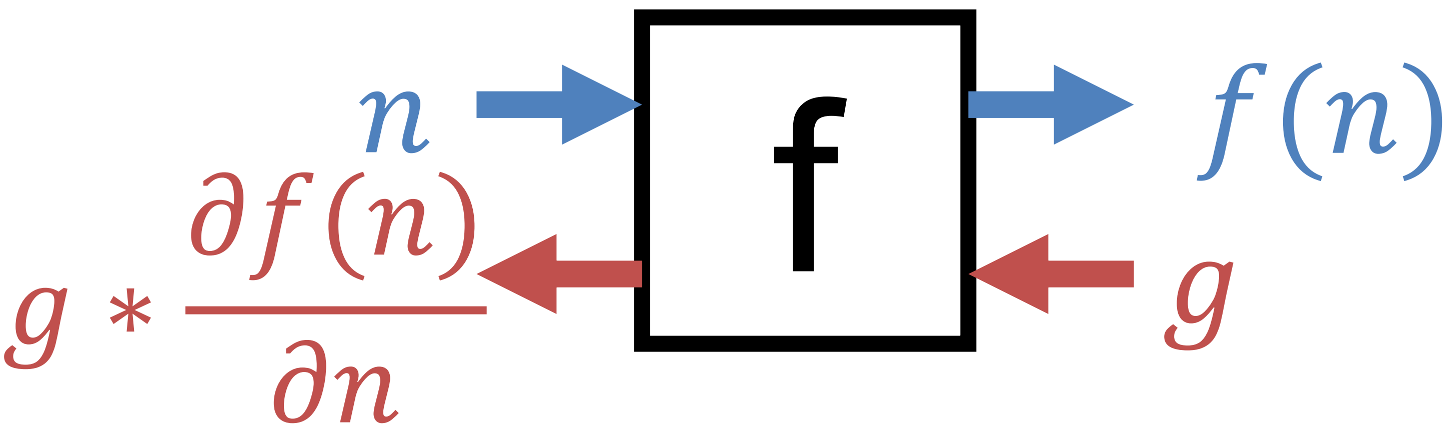
Let's Do This Another Way

Suppose we have a box representing a function f.

This box does two things:

Forward: Given forward input **n**, compute **f(n)**

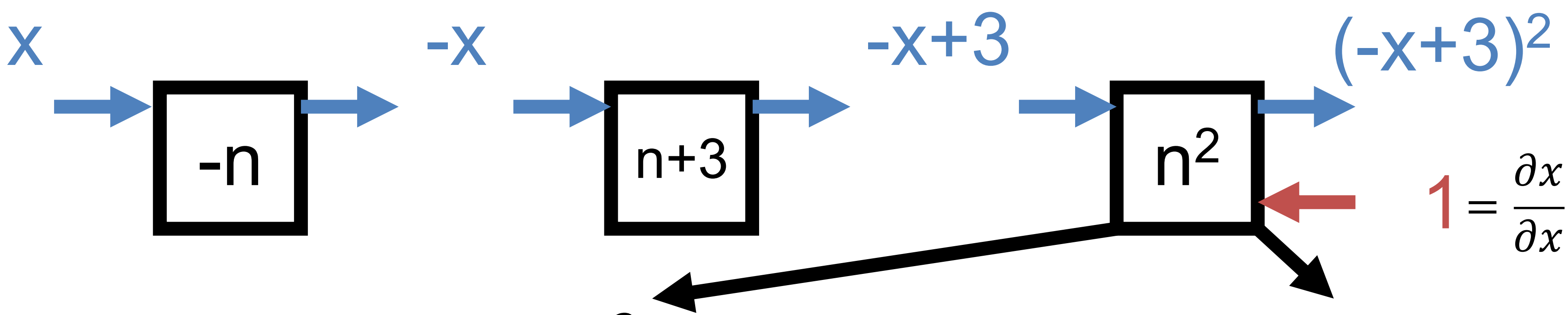
Backwards: Given backwards input **g**, return **g*df/dn**



g*df/dn

Let's Do This Another Way

$$f(x) = (-x + 3)^2$$



$$\frac{\partial}{\partial n} n^2 = 2n = 2(-x + 3)$$

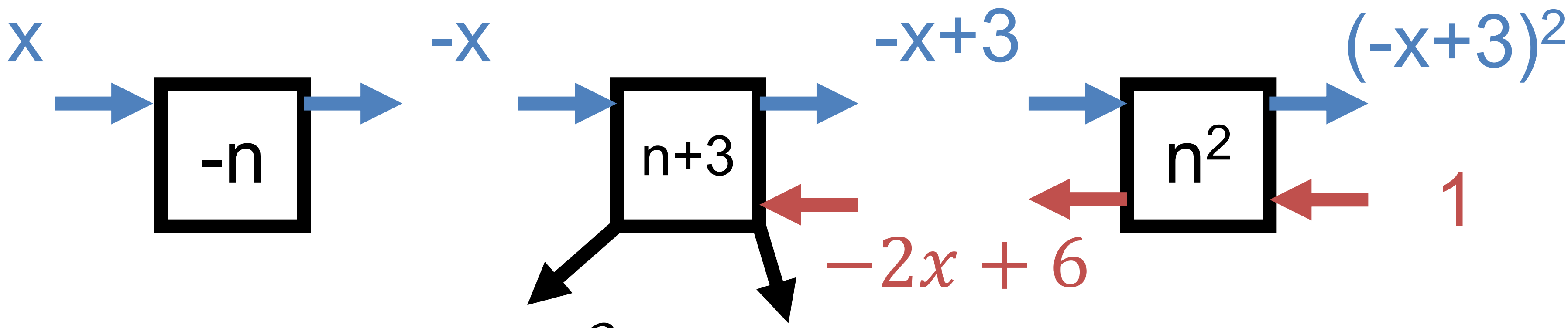
$$= -2x + 6$$

$$\frac{\partial}{\partial n} * 1 = (-2x + 6) * 1$$

g*df/dn

Let's Do This Another Way

$$f(x) = (-x + 3)^2$$



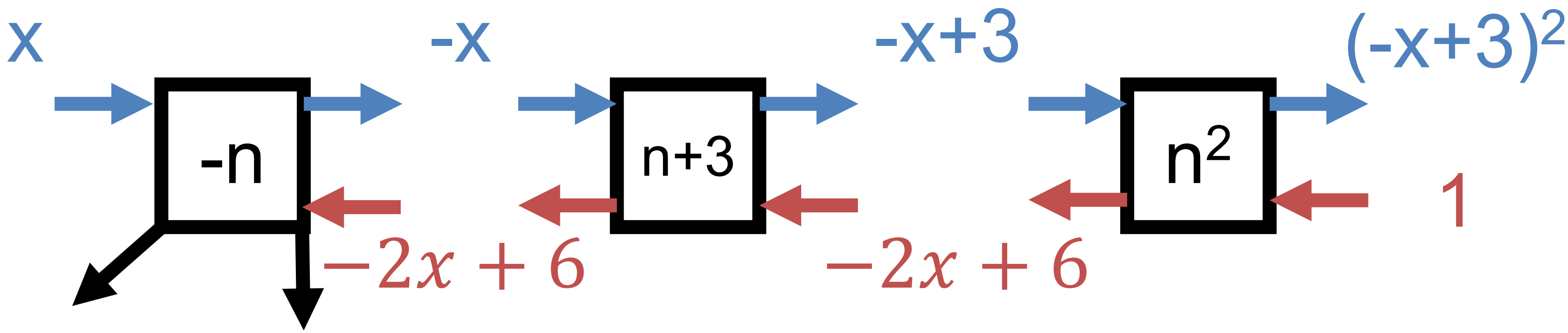
$$\frac{\partial}{\partial n} = 1$$

$$1 * (-2x + 6)$$

g*df/dn

Let's Do This Another Way

$$f(x) = (-x+3)^2$$



$$\frac{\partial}{\partial n} = -1$$

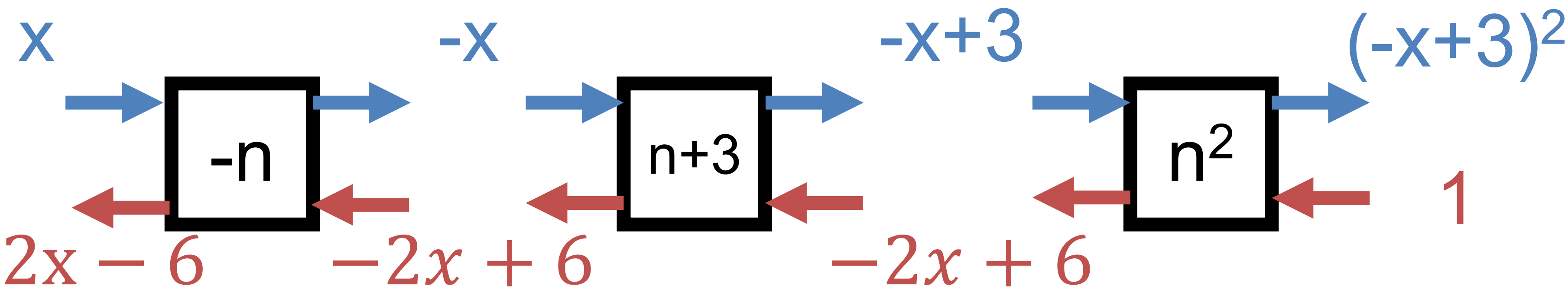
$$-1 \; * \; (-2\mathrm{x}+6)$$

$$2\mathrm{x}-6$$

g*df/dn

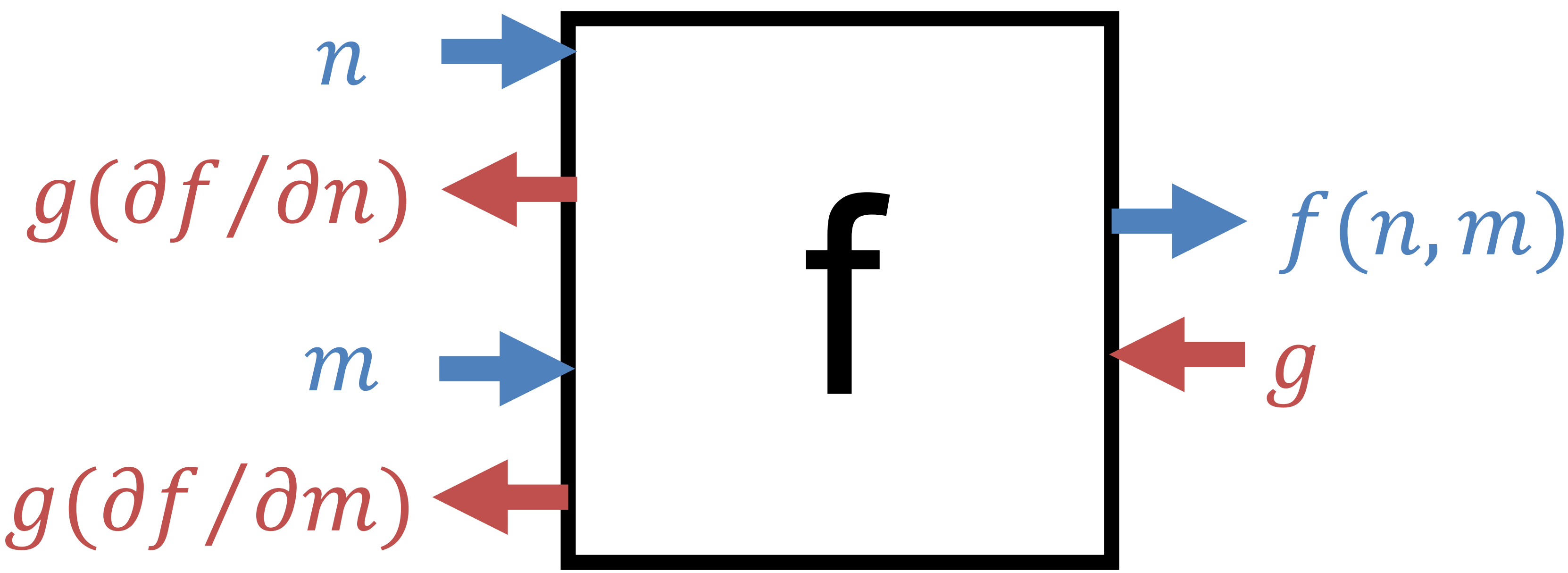
Let's Do This Another Way

$$f(x) = (-x+3)^2$$

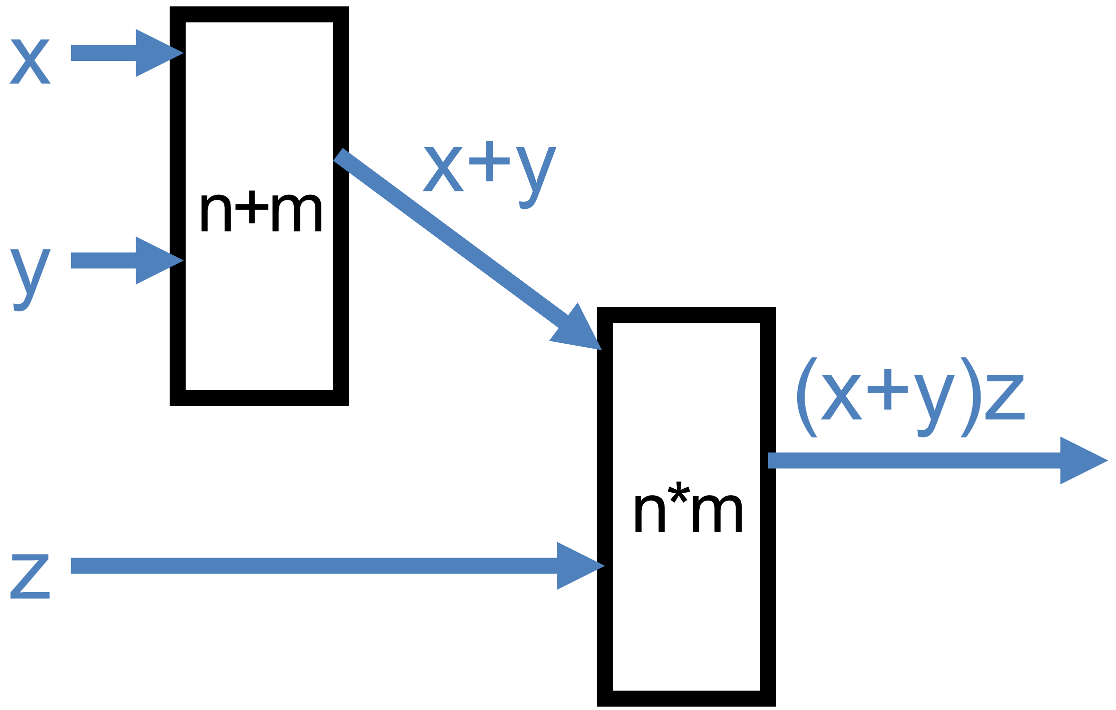


Two Inputs

Given two inputs, just have two input/output wires
Forward: the same
Backward: the same – send gradients with respect to each variable



$$f(x,y,z) = (x+y)z$$

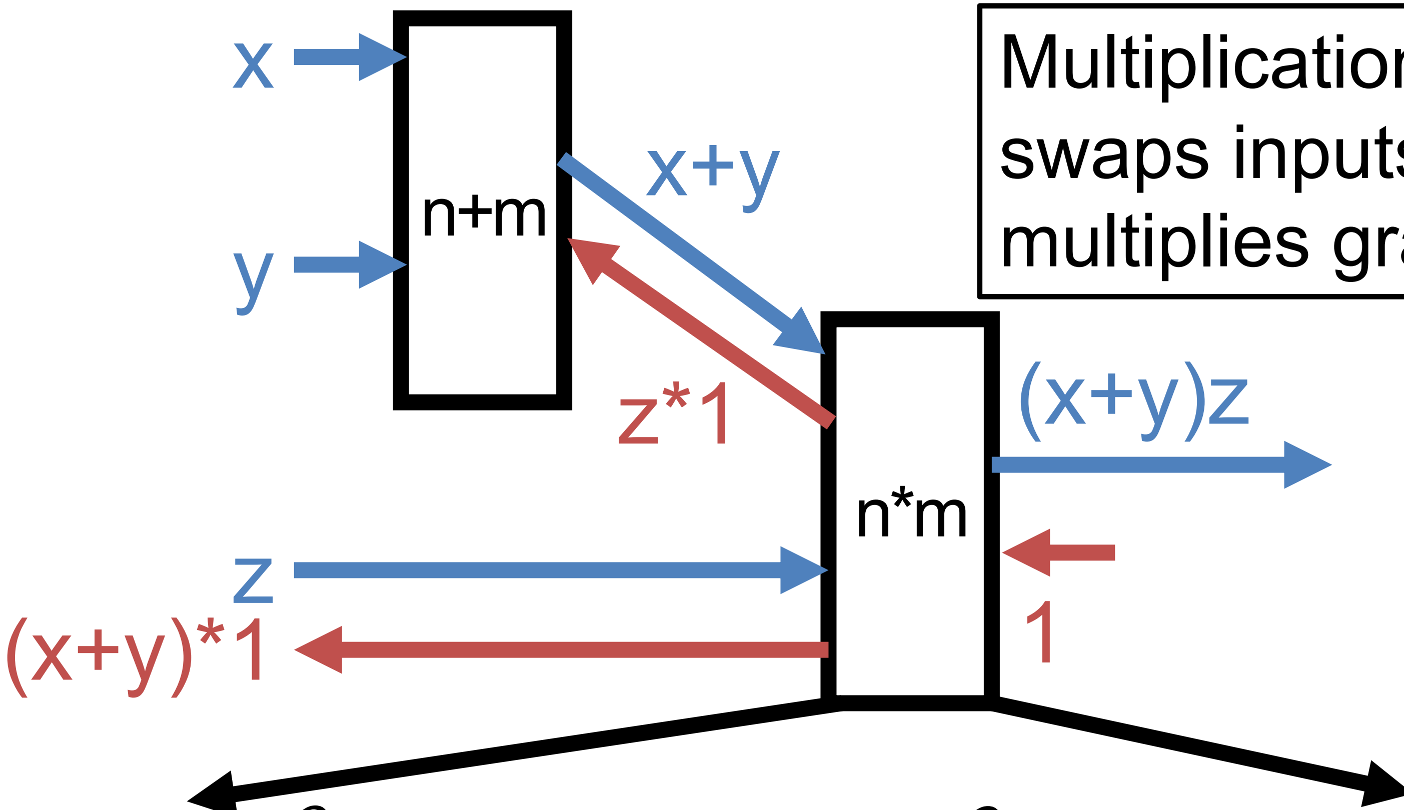


Example Credit: Karpathy and Fei-Fei

g*df/dn

$f(x,y,z) = (x+y)z$

Multiplication swaps inputs, multiplies gradient



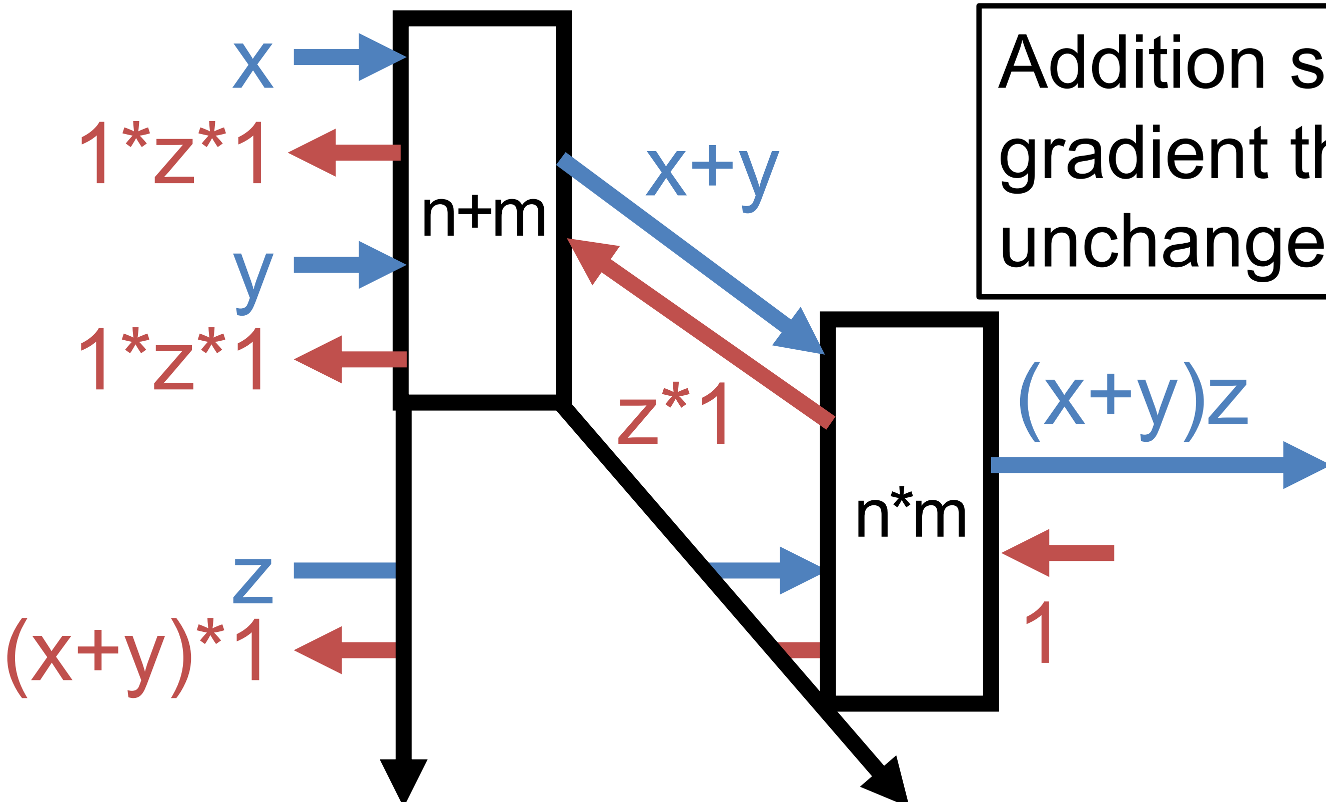
$$\frac{\partial}{\partial n} nm = m$$
$$\rightarrow z * 1$$

$$\frac{\partial}{\partial m} nm = n$$
$$\rightarrow (x+y) * 1$$

Example Credit: Karpathy and Fei-Fei

g*df/dn

$$f(x,y,z) = (x+y)z$$



Addition sends gradient through unchanged

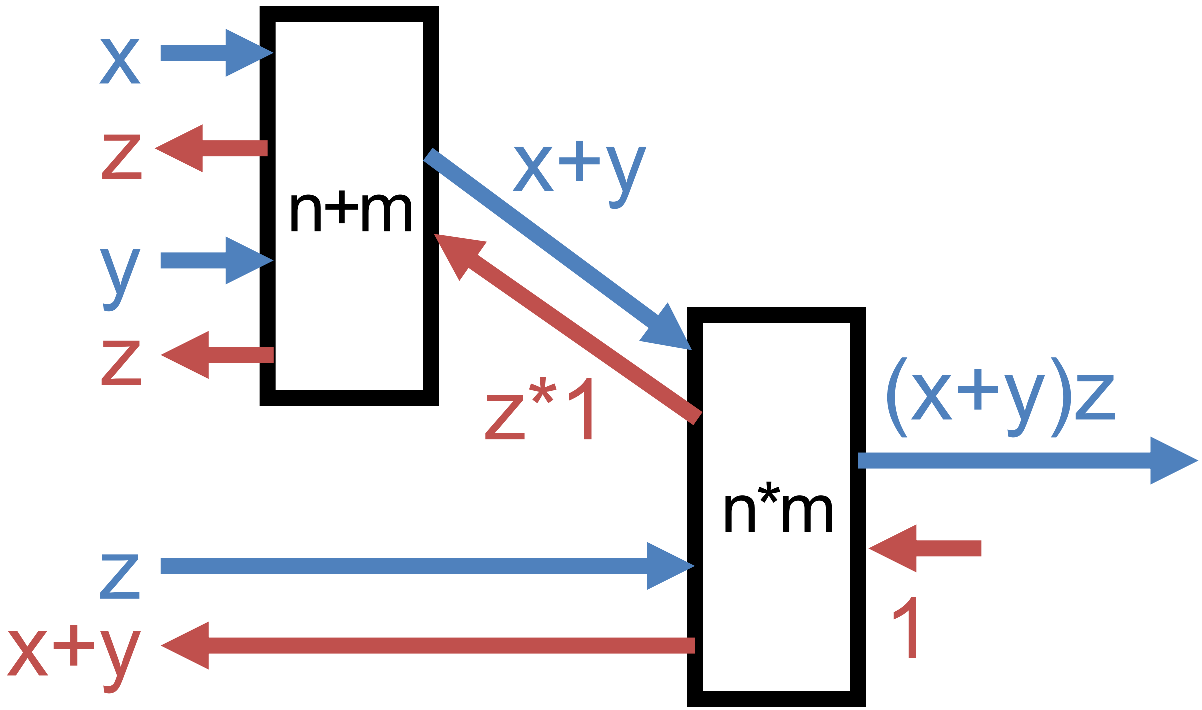
$$\frac{\partial}{\partial n} n + m = 1 \qquad \frac{\partial}{\partial m} n + m = 1$$

$$\rightarrow 1 * z * 1 \qquad \rightarrow 1 * z * 1$$

Example Credit: Karpathy and Fei-Fei

g*df/dn

f(x,y,z) = (x+y)z

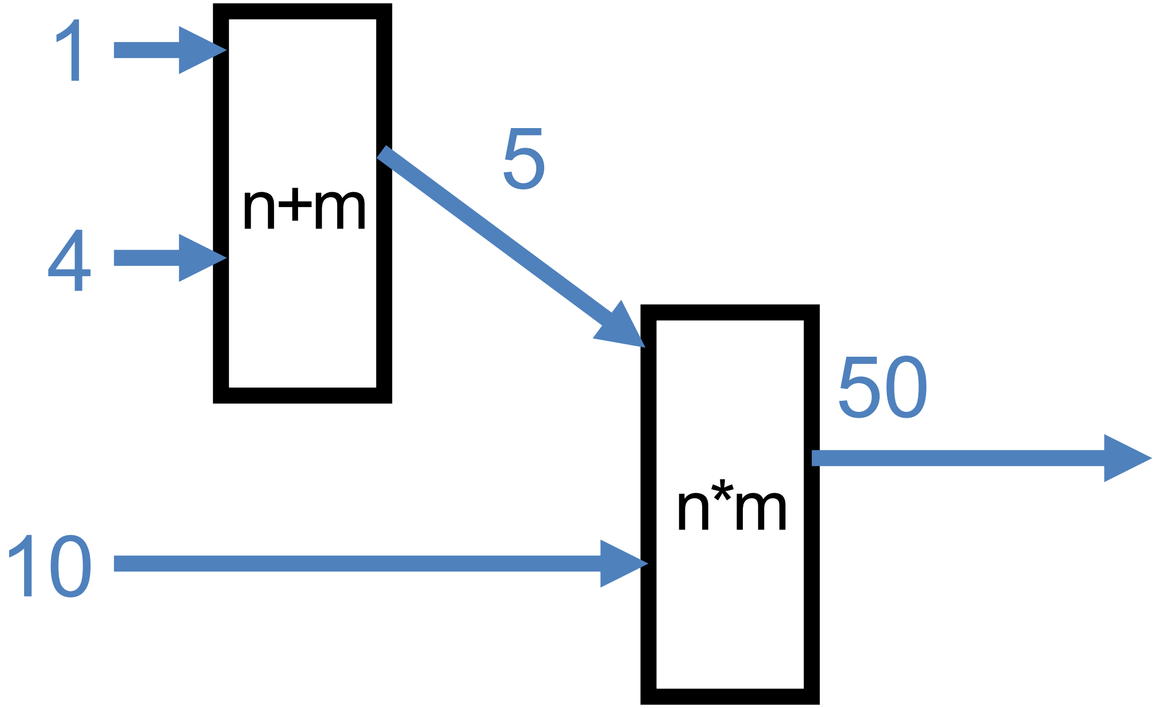


$$\frac{\partial(x+y)z}{\partial x} = z \quad \frac{\partial(x+y)z}{\partial y} = z \quad \frac{\partial(x+y)z}{\partial z} = (x+y)$$

Example Credit: Karpathy and Fei-Fei

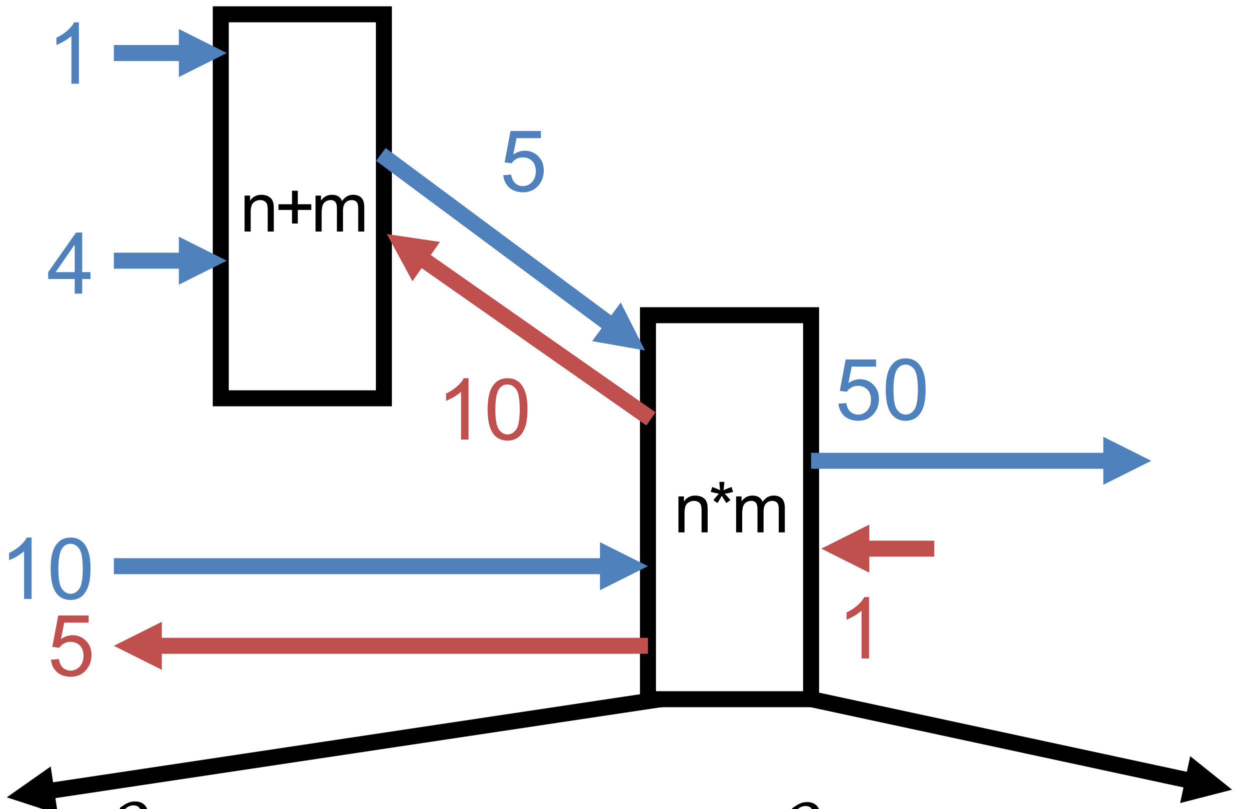
Once More, With Numbers!

$$f(x,y,z) = (x+y)z$$



Example Credit: Karpathy and Fei-Fei

f(x,y,z) = (x+y)z



$$\frac{\partial}{\partial n} nm = m$$

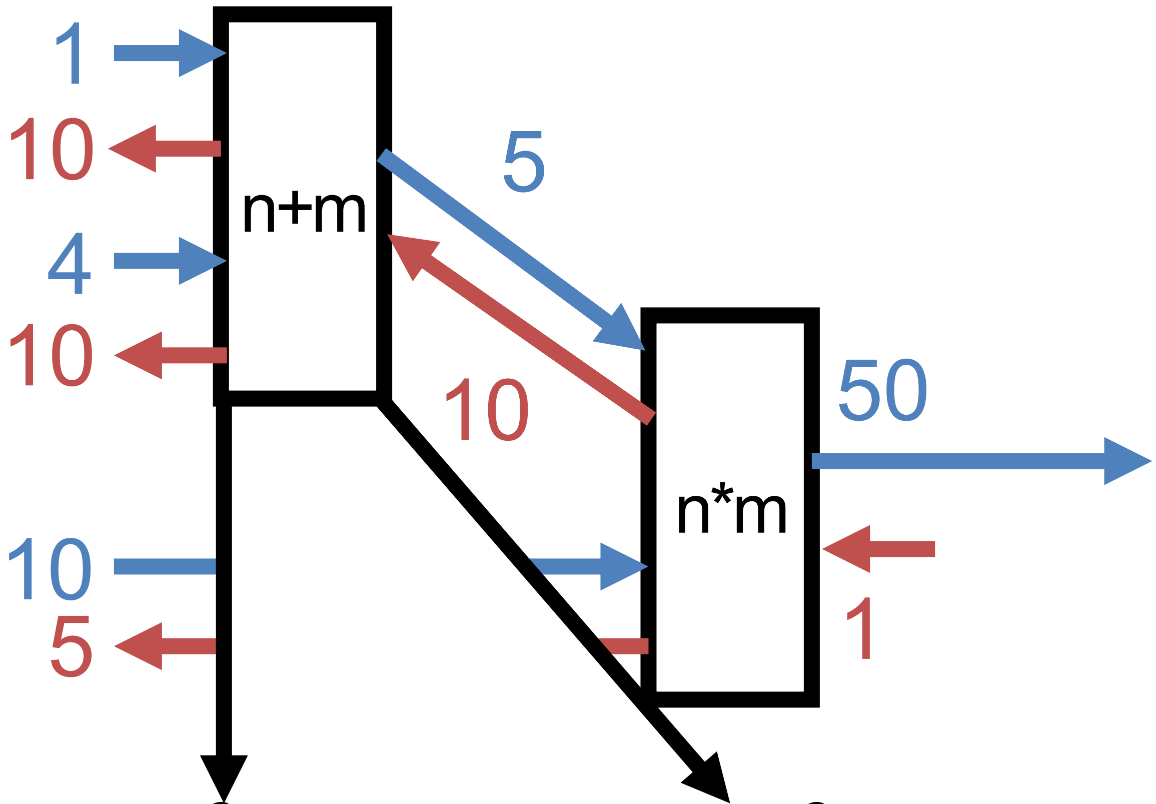
$$\rightarrow 10 * 1$$

$$\frac{\partial}{\partial m} nm = n$$

$$\rightarrow 5 * 1$$

Example Credit: Karpathy and Fei-Fei

f(x,y,z) = (x+y)z



$$\frac{\partial}{\partial n} n + m = 1 \qquad \frac{\partial}{\partial m} n + m = 1$$

$$\rightarrow 1 * 10 * 1 \qquad \rightarrow 1 * 10 * 1$$

Example Credit: Karpathy and Fei-Fei

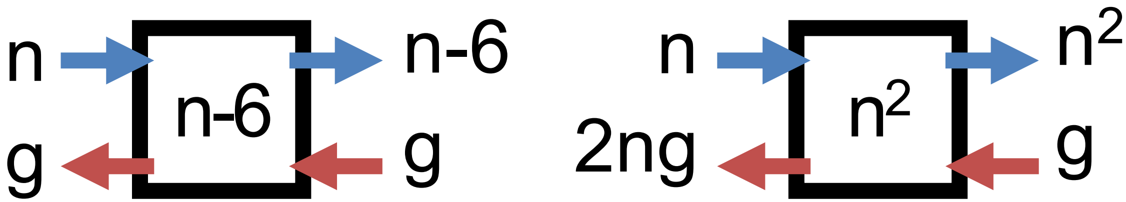
Think You've Got It?

$$L(x) = (w - 6)^2$$

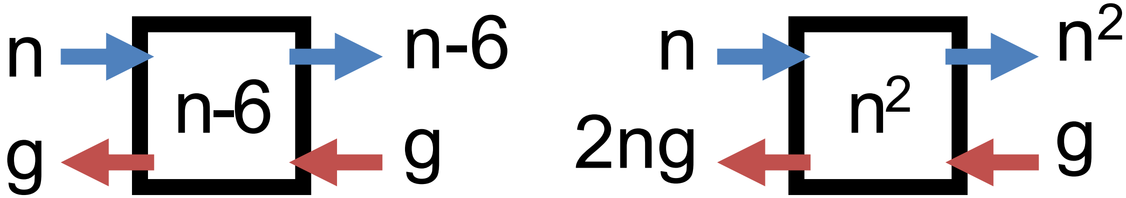
- We want to fit a model w that just will equal 6.
- World's most basic linear model / neural net: no inputs, just constant output.

I'll Need a Few Volunteers

$$L(x) = (w-6)^2$$

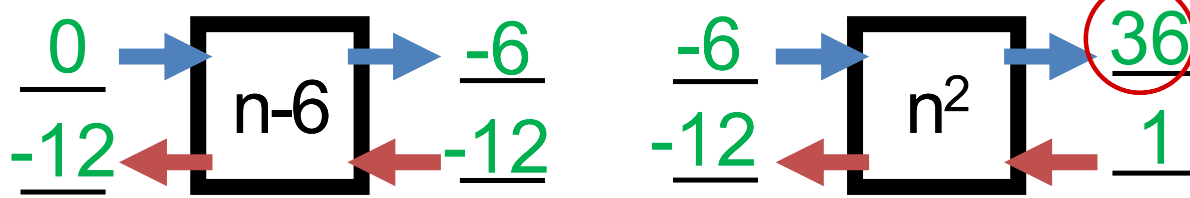


Job #1 (n-6): **Forward:** Compute n-6 **Backwards:** Multiply by 1	Job #2 (n^2): **Forward:** Compute n^2 **Backwards:** Multiply by 2n	Job #3: **Backwards:** Give me a 1



$$L(x) = (w-6)^2$$

w0: 0



this is your loss

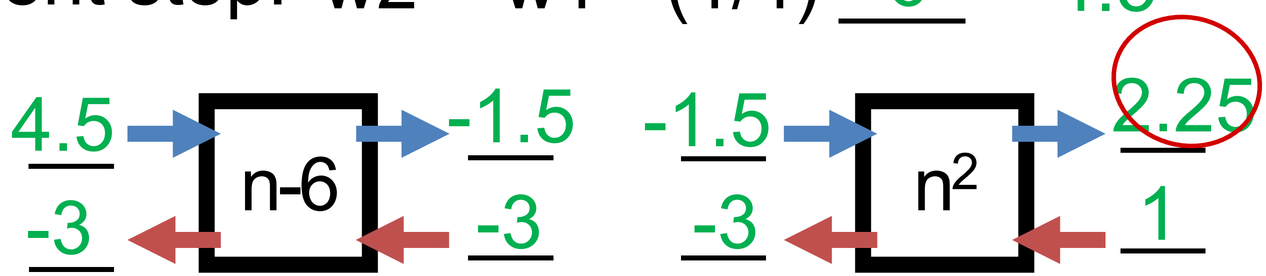
Gradient step: w1 = w0 - (1/4) -12 = 3

loss decreasing

w1: 3

Gradient step: w2 = w1 - (1/4) -6 = 4.5

w2: 4.5



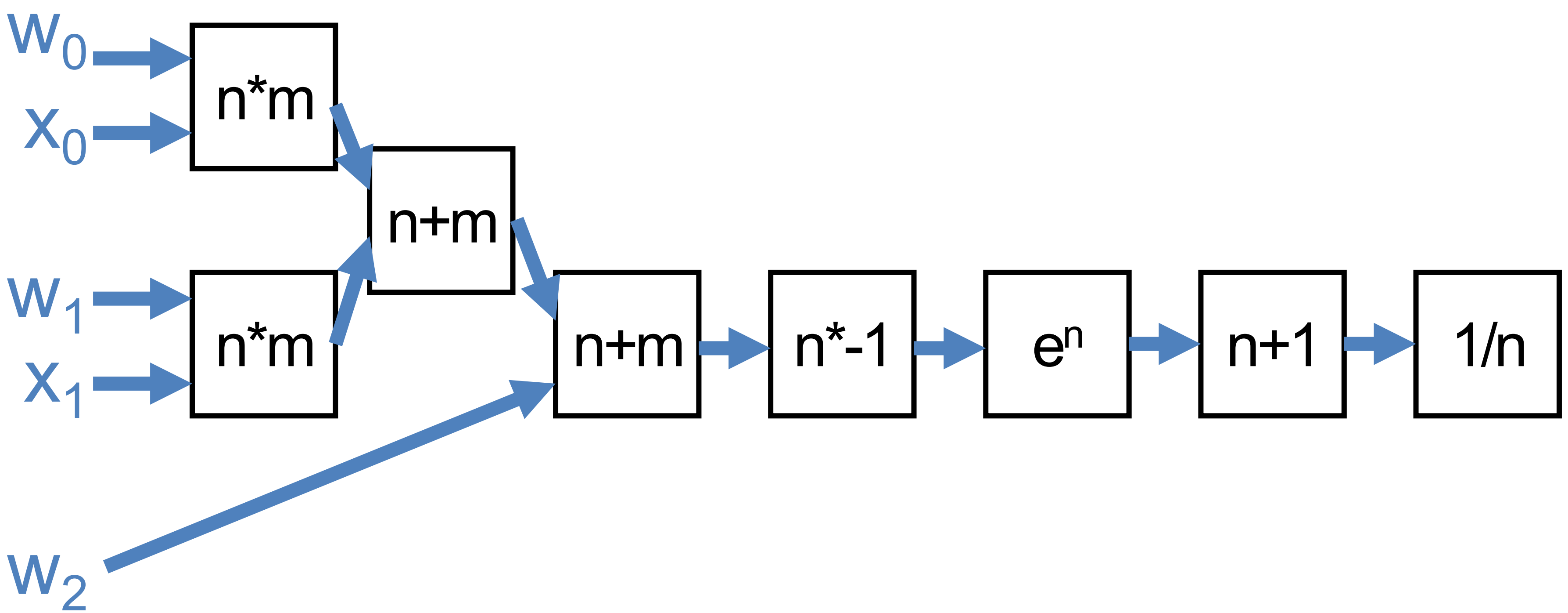
Gradient step: w3 = w2 - (1/4) -3 = 5.25

Preemptively

- The diagrams look complex but that’s since we’re covering the details together

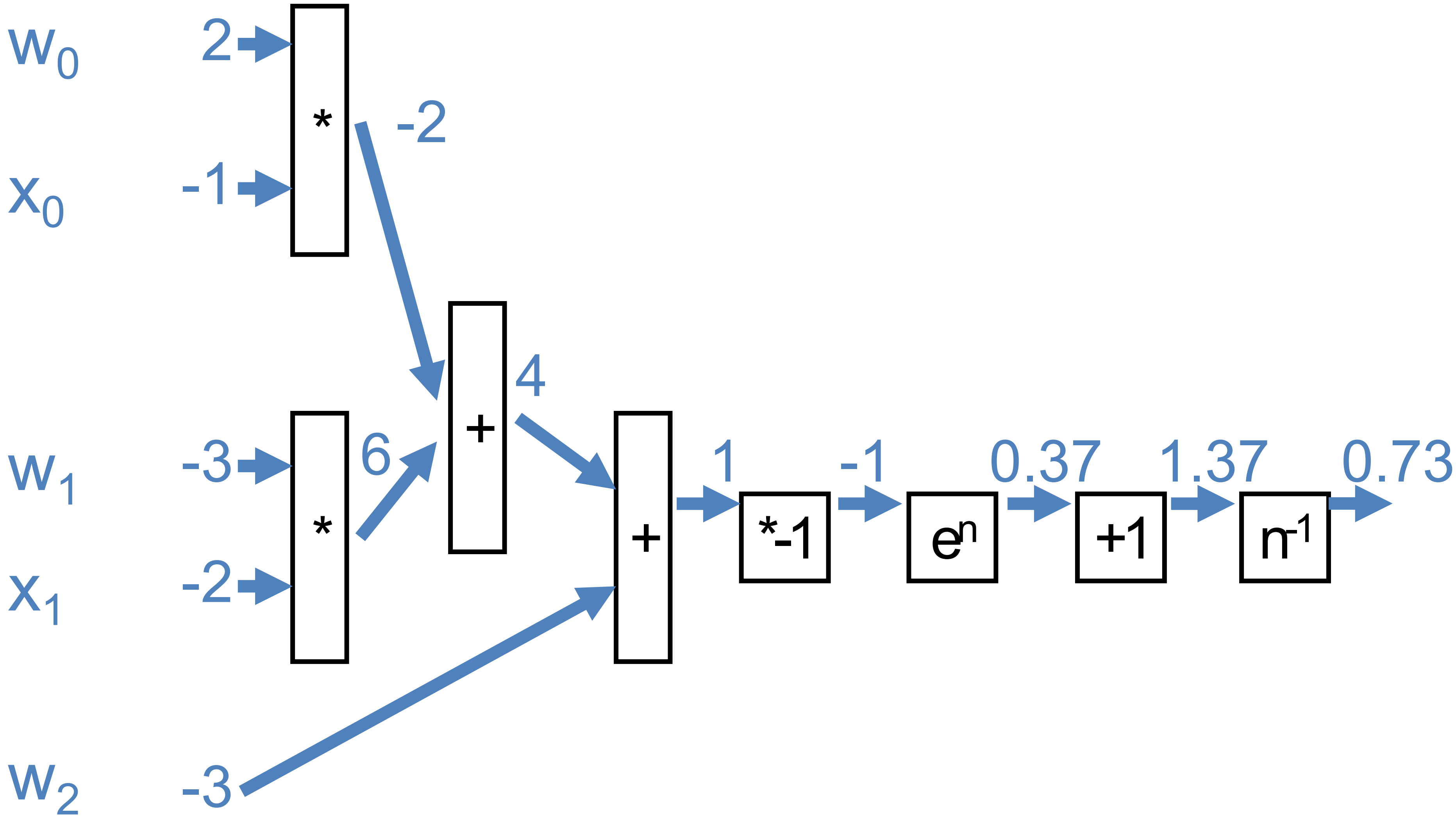
Something More Complex

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

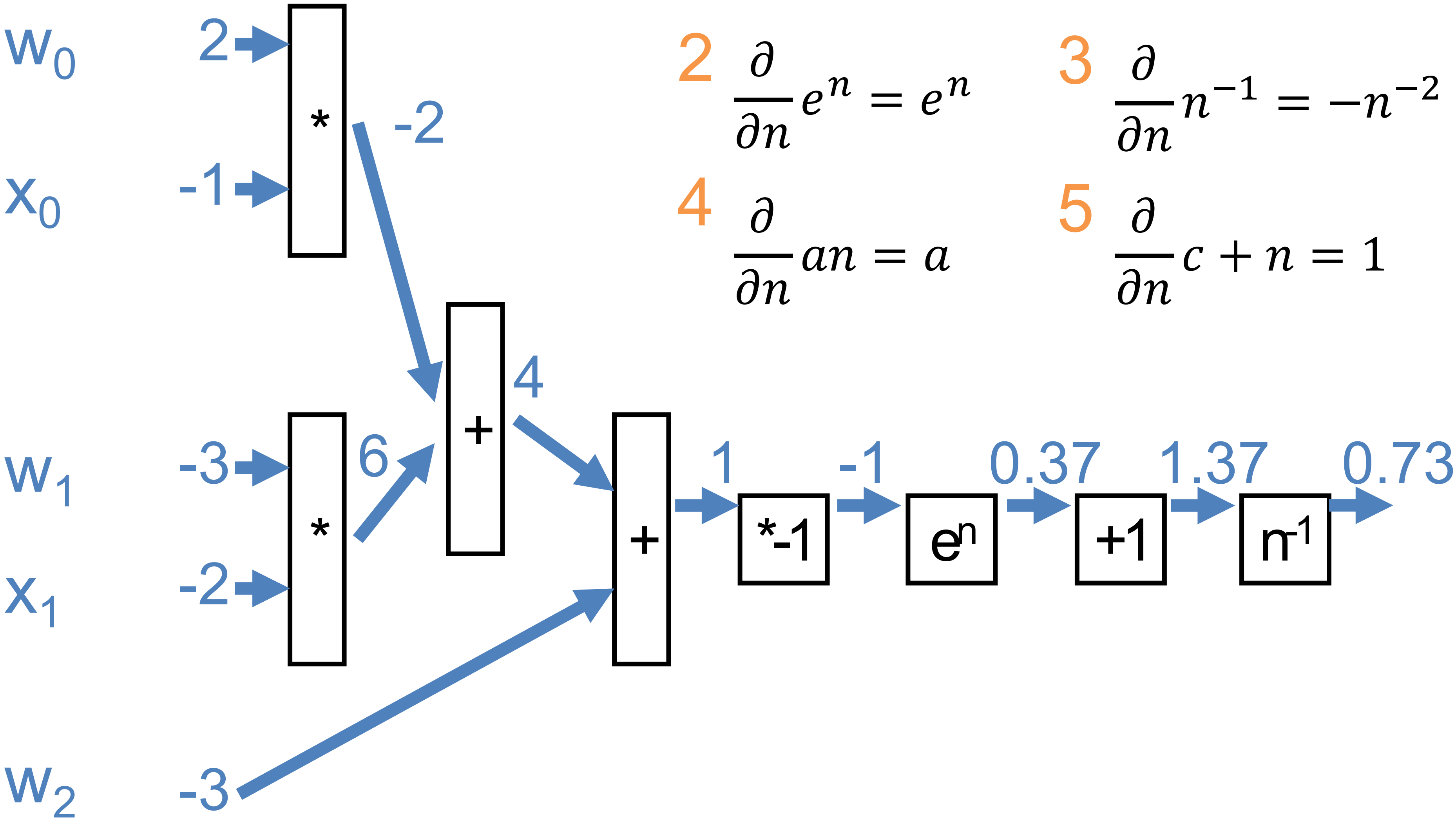
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



Example Credit: Karpathy and Fei-Fei

$$f(\mathbf{w}, \mathbf{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\dfrac{\partial}{\partial n} m + n = 1$

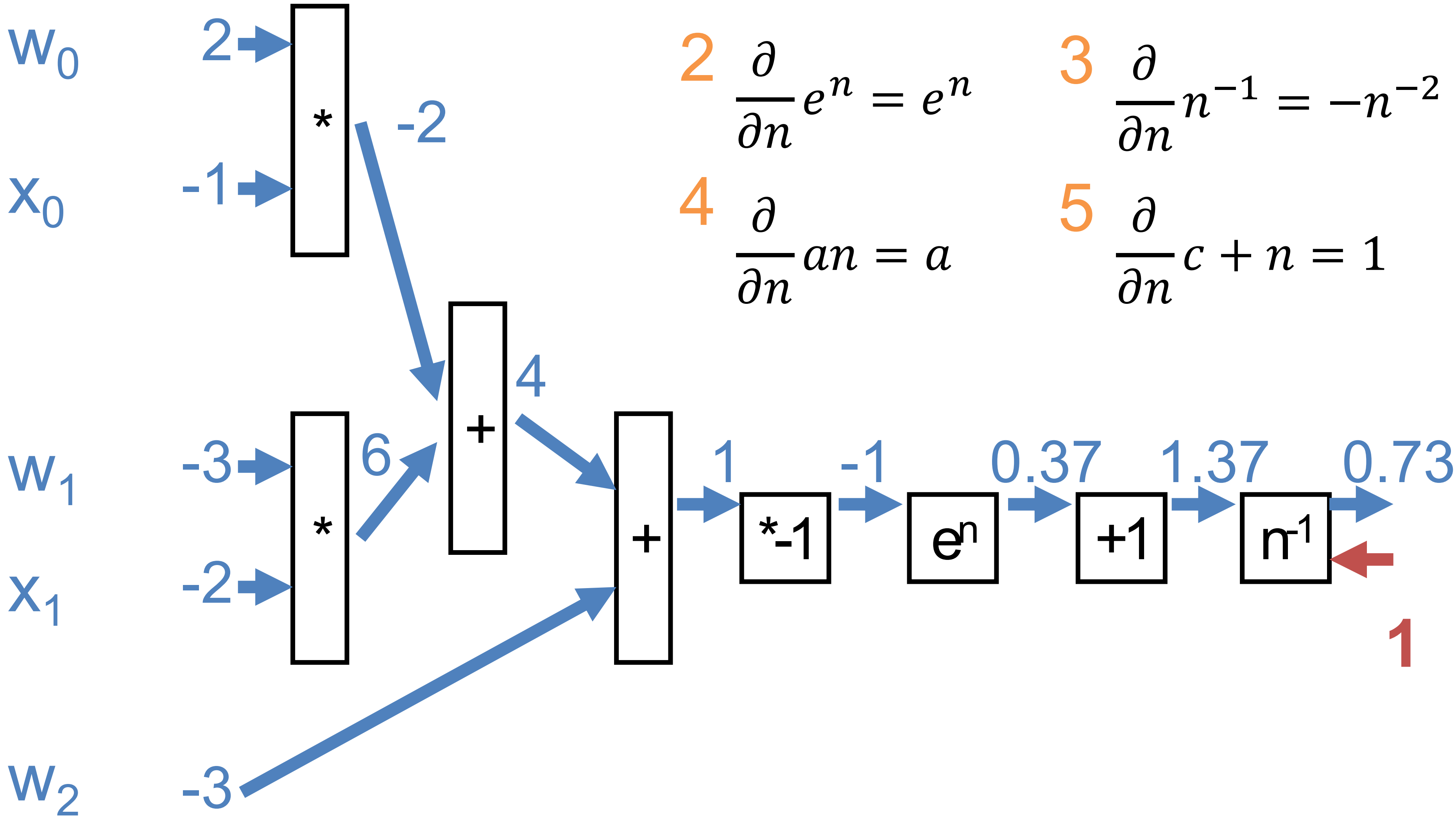
1 $\dfrac{\partial}{\partial n} mn = m$

2 $\dfrac{\partial}{\partial n} e^n = e^n$

3 $\dfrac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\dfrac{\partial}{\partial n} an = a$

5 $\dfrac{\partial}{\partial n} c + n = 1$



Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

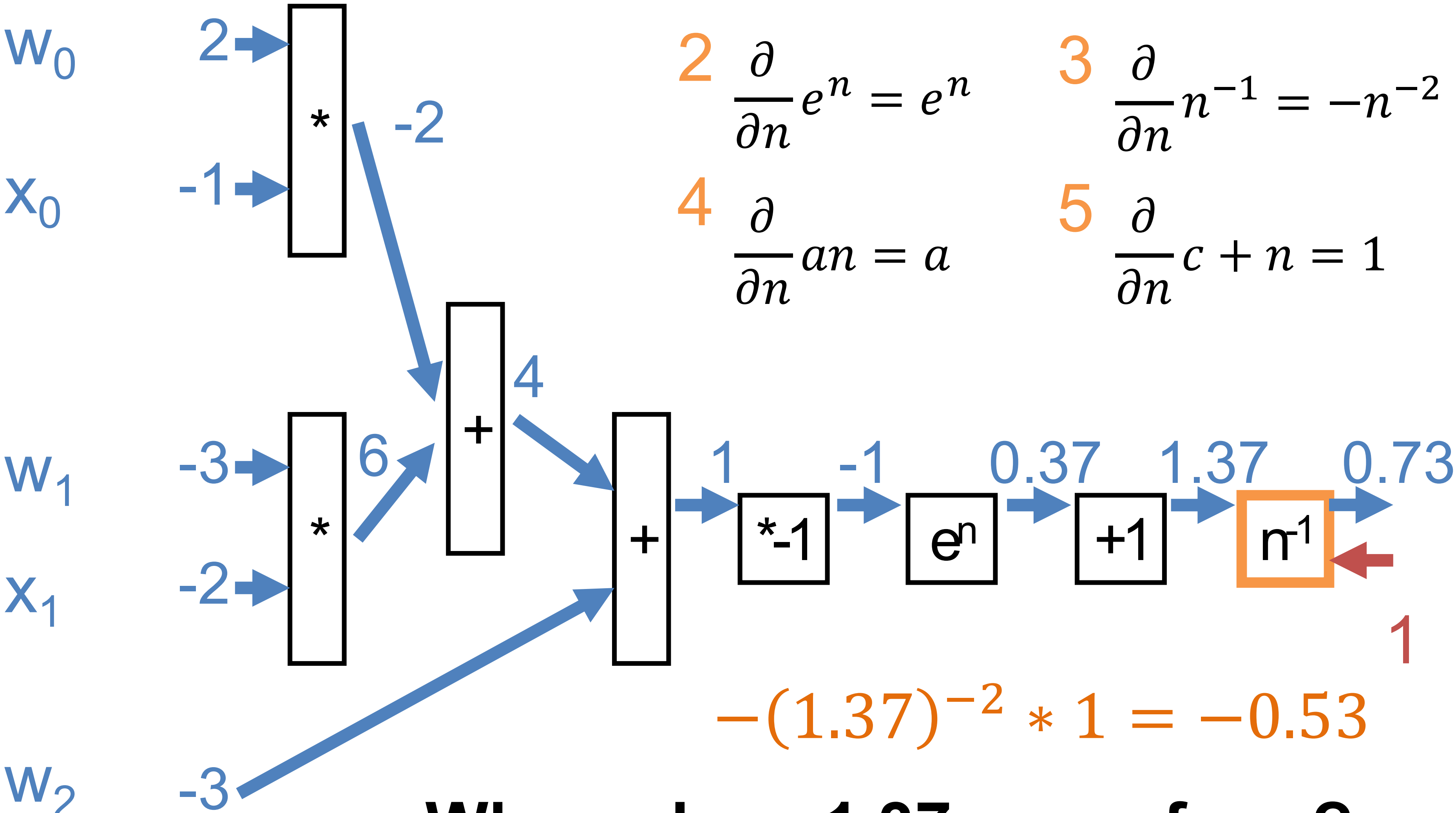
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



$$-(1.37)^{-2} * 1 = -0.53$$

Where does 1.37 come from?

Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

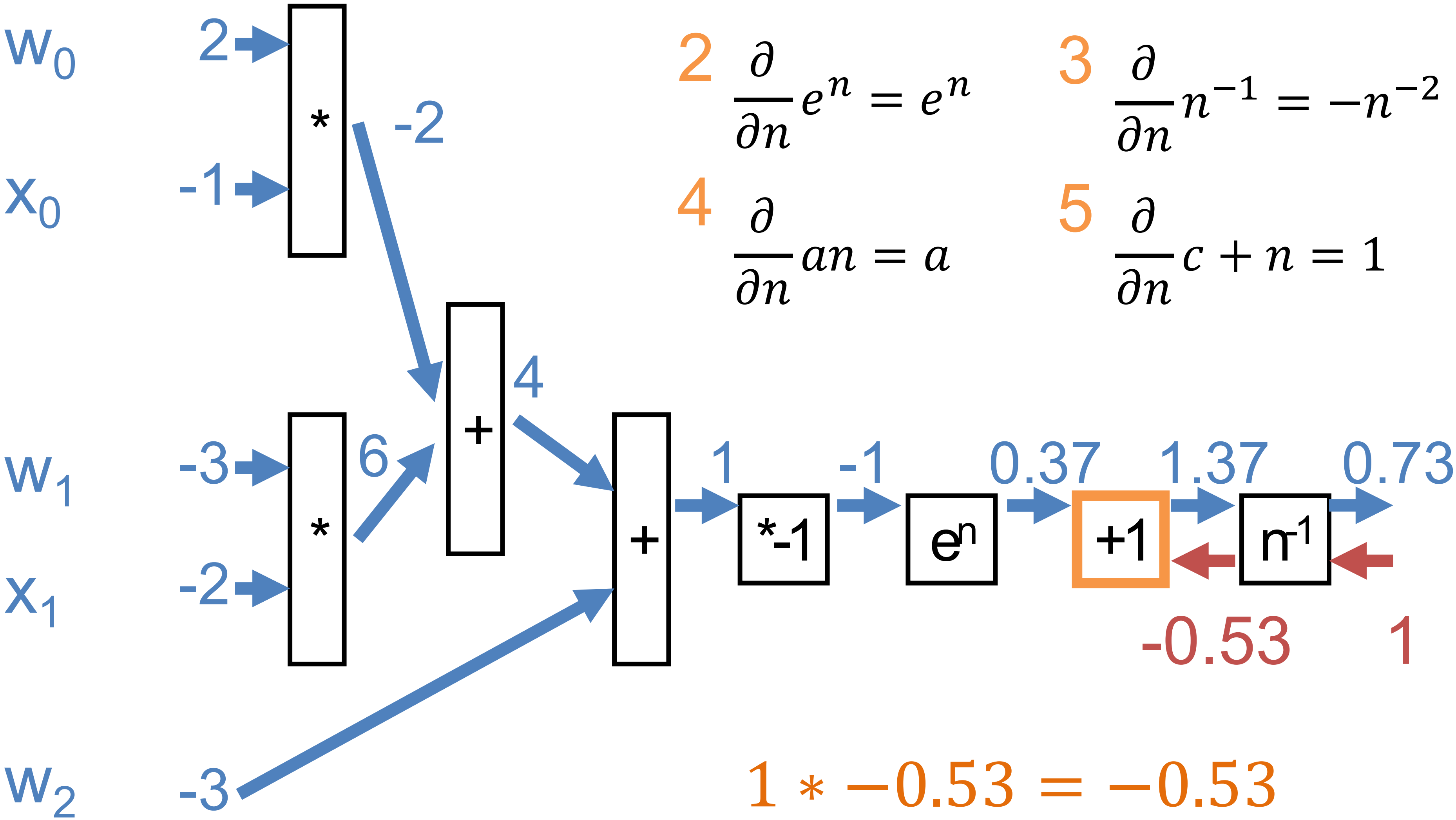
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



$$1 * -0.53 = -0.53$$

Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

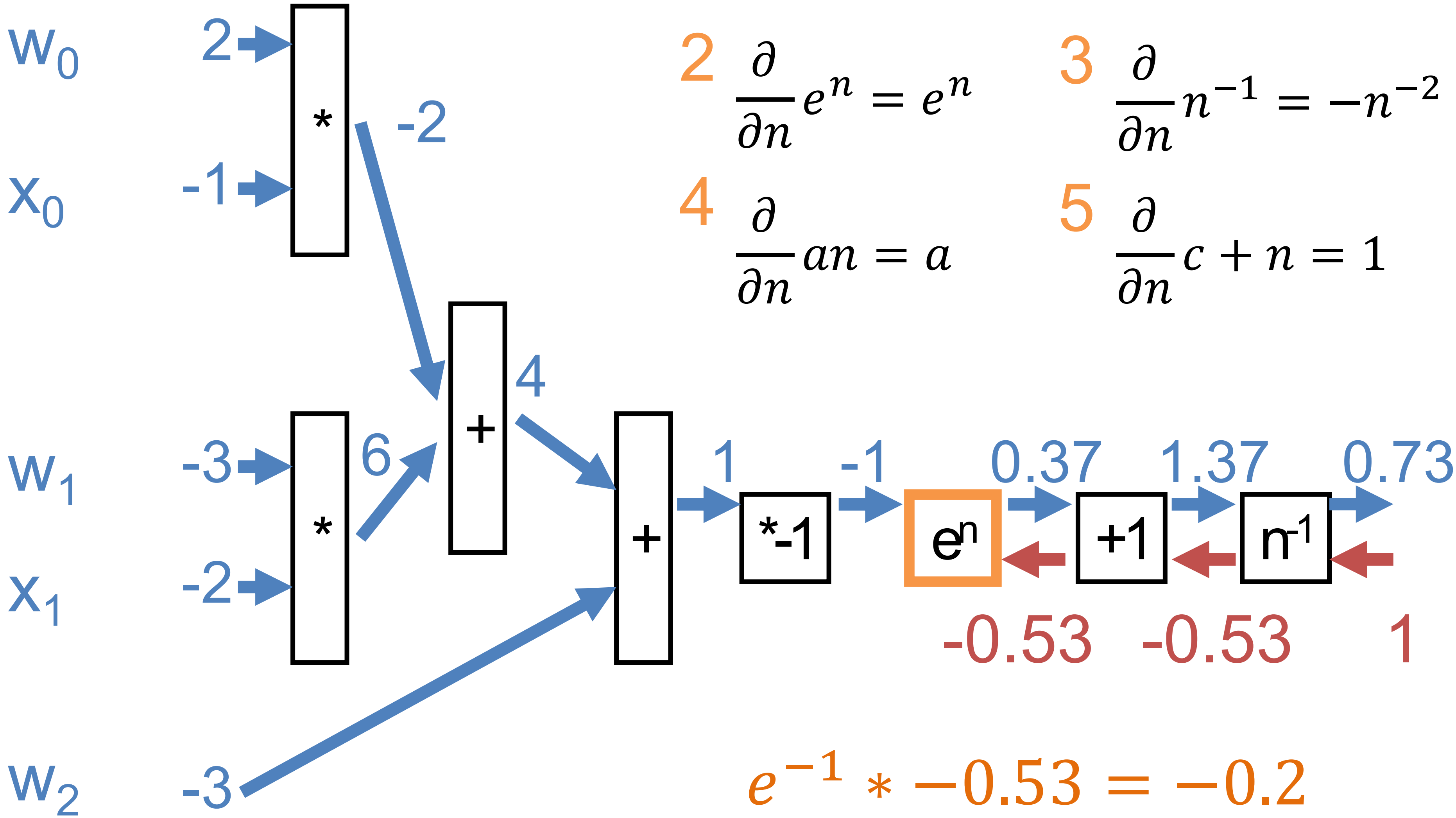
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



$$e^{-1} * -0.53 = -0.2$$

Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

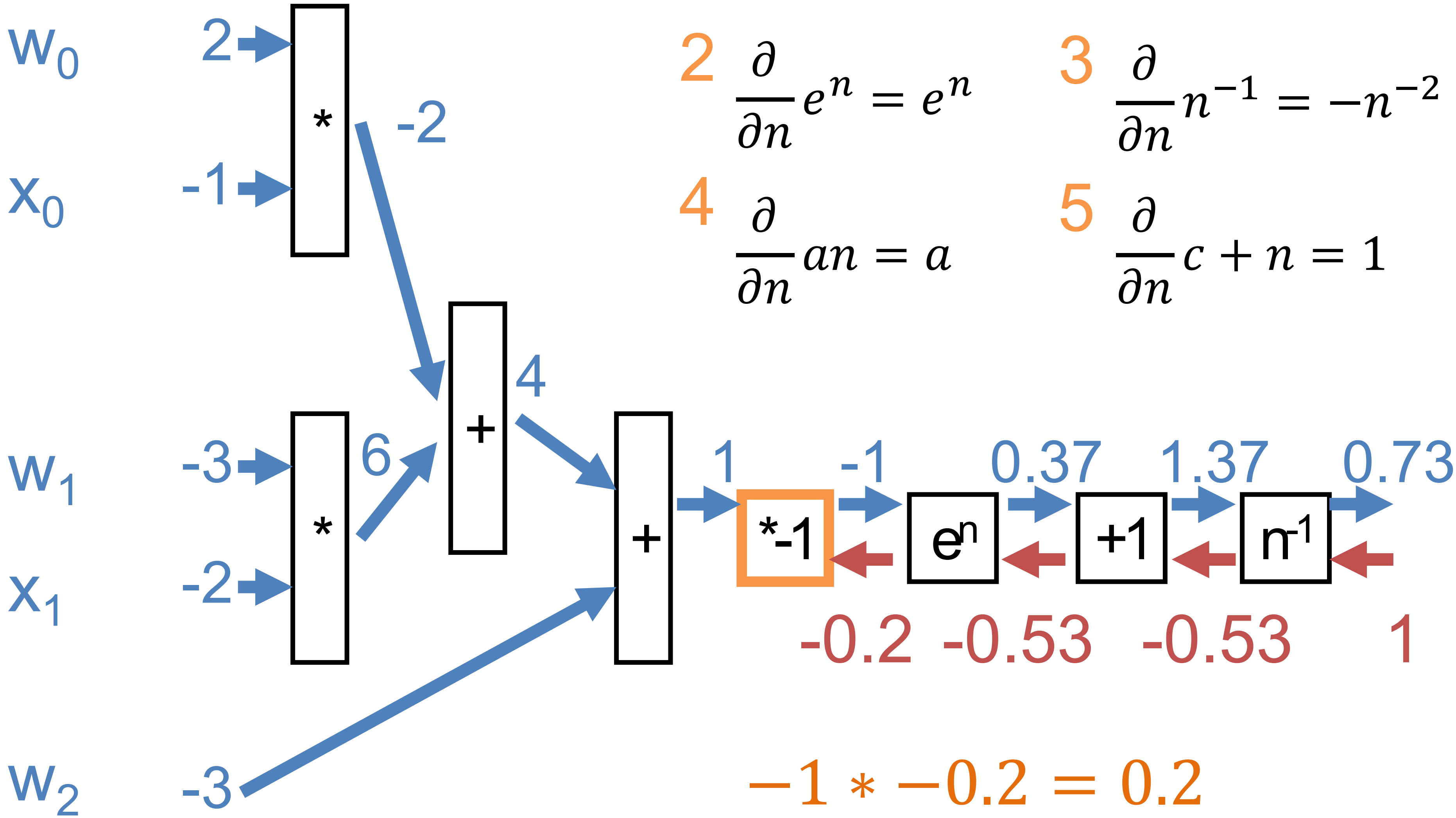
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



$$-1 * -0.2 = 0.2$$

Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

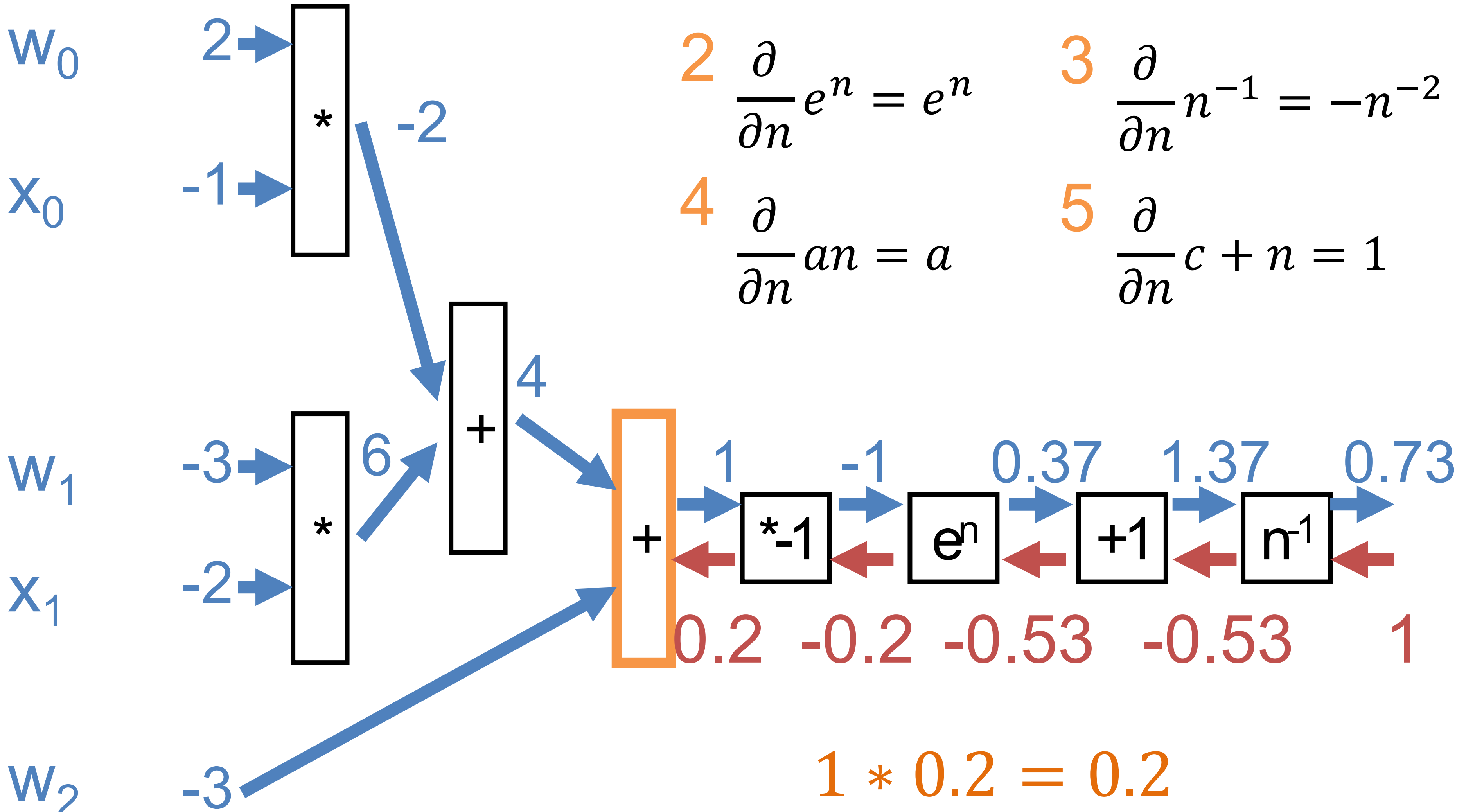
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



$1 * 0.2 = 0.2$

Gets sent back both directions

Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

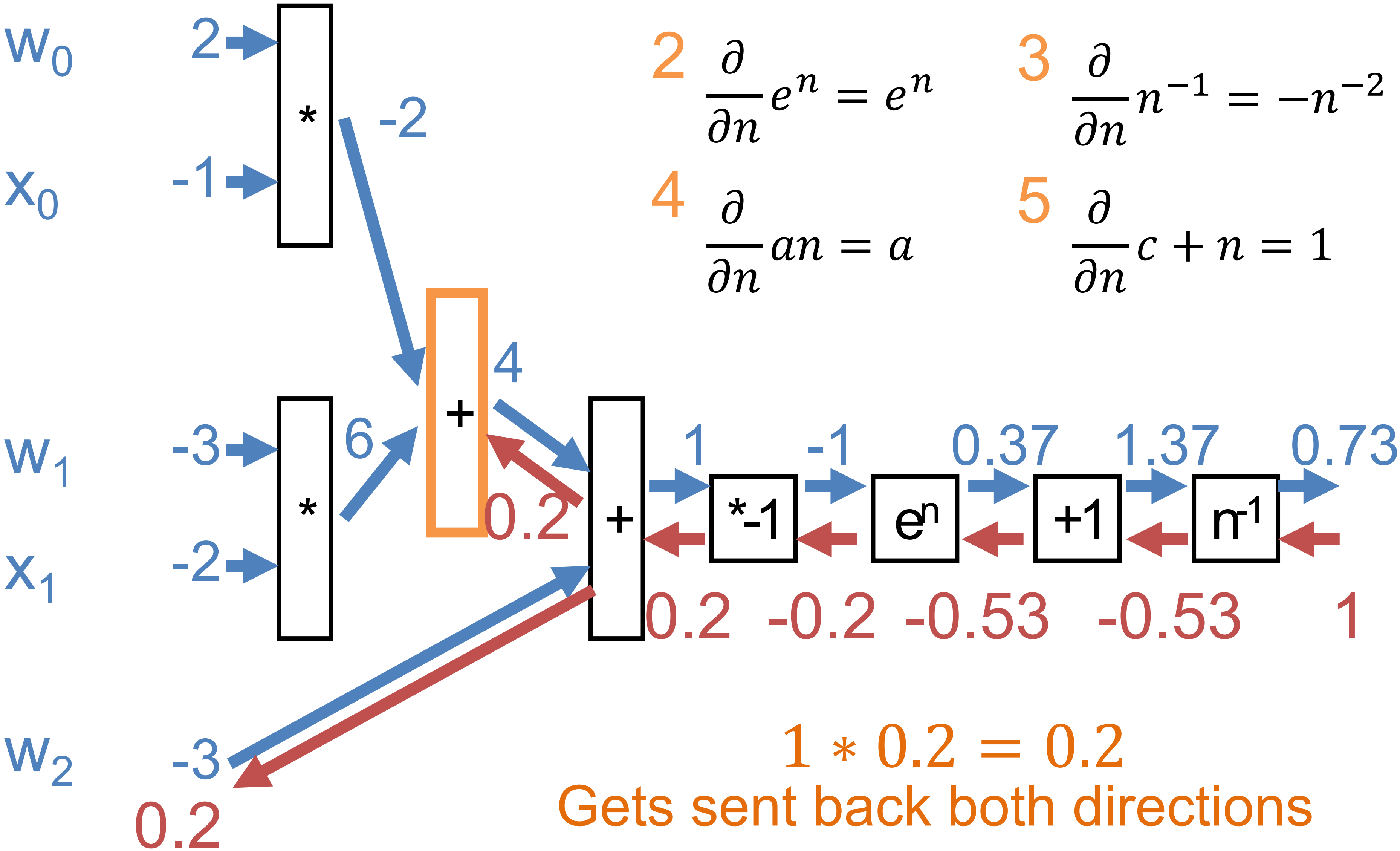
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



$1 * 0.2 = 0.2$

Gets sent back both directions

Example Credit: Karpathy and Fei-Fei

$$f(\mathbf{w}, \mathbf{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

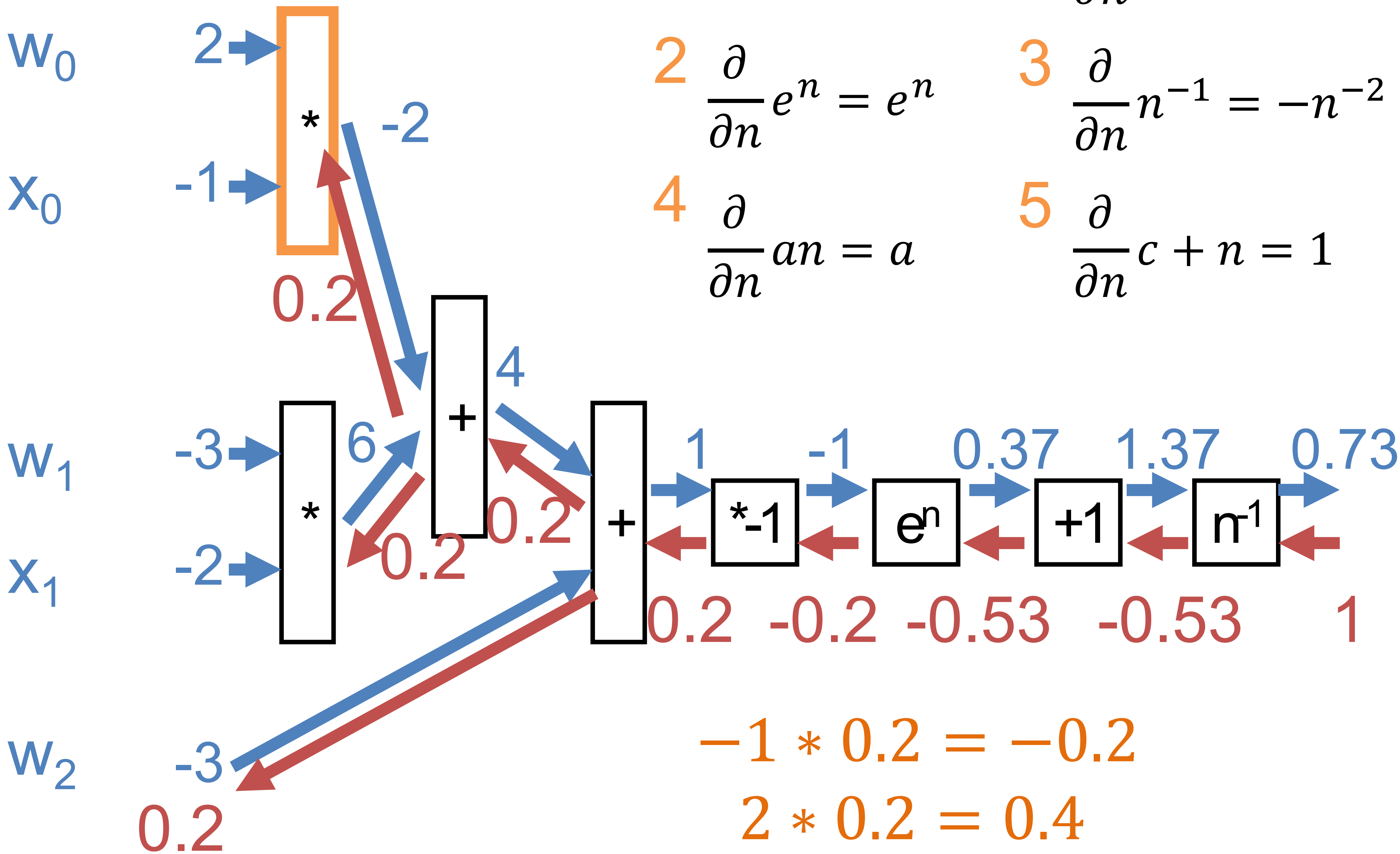
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



$$-1 * 0.2 = -0.2$$

$$2 * 0.2 = 0.4$$

Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

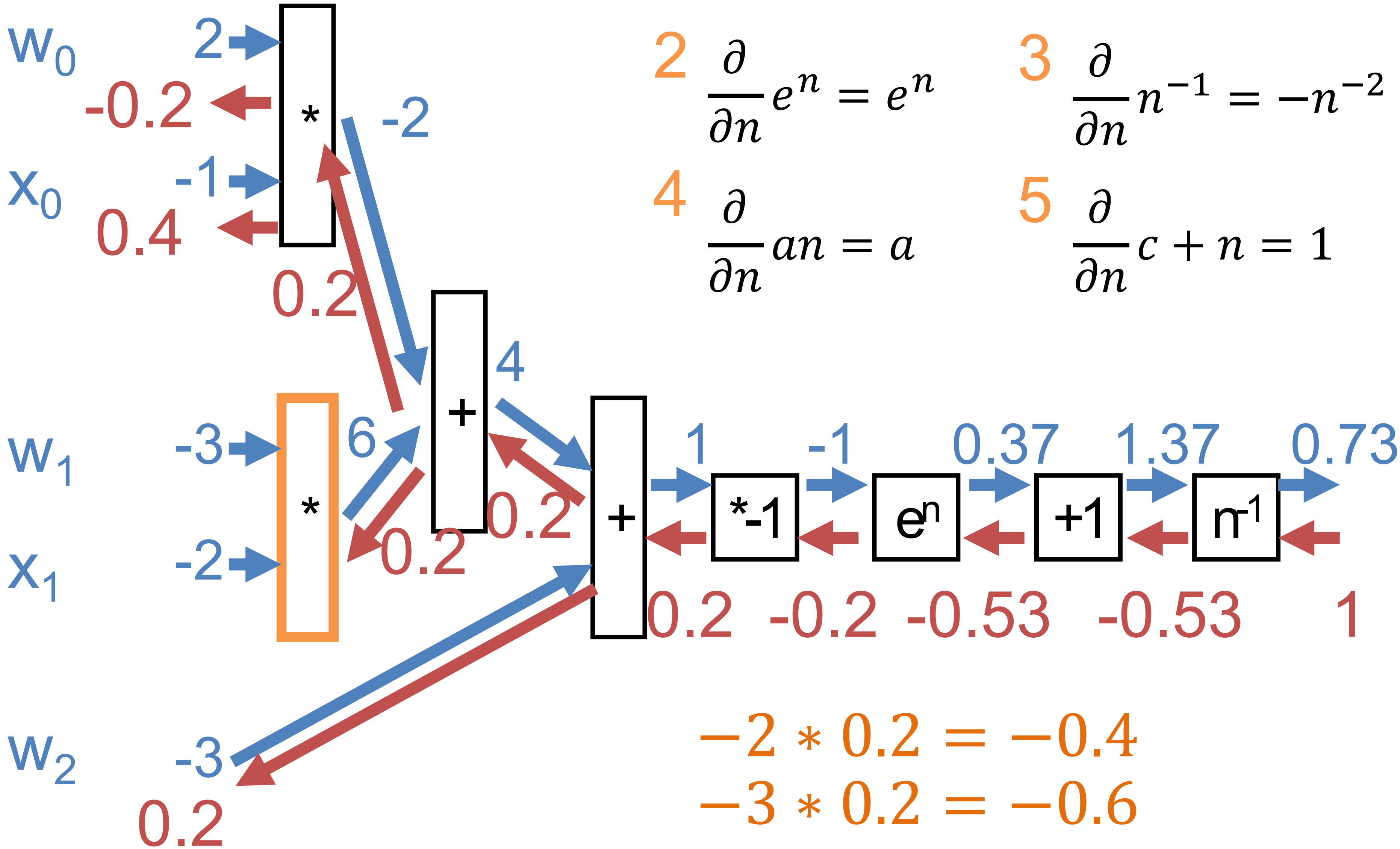
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



$$-2 * 0.2 = -0.4$$
$$-3 * 0.2 = -0.6$$

Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\frac{\partial}{\partial n} m + n = 1$

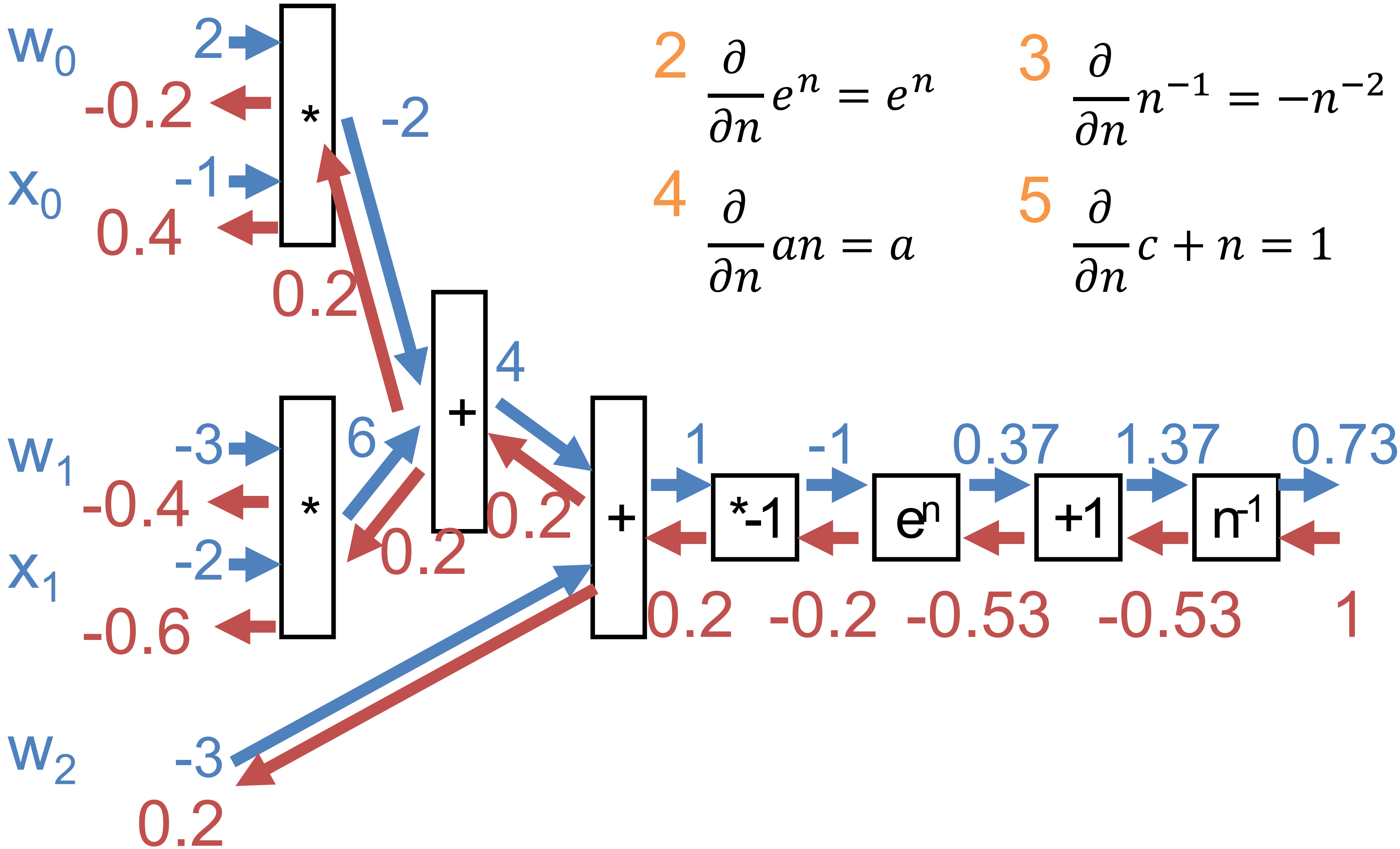
1 $\frac{\partial}{\partial n} mn = m$

2 $\frac{\partial}{\partial n} e^n = e^n$

3 $\frac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\frac{\partial}{\partial n} an = a$

5 $\frac{\partial}{\partial n} c + n = 1$



Example Credit: Karpathy and Fei-Fei

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

0 $\dfrac{\partial}{\partial n} m + n = 1$

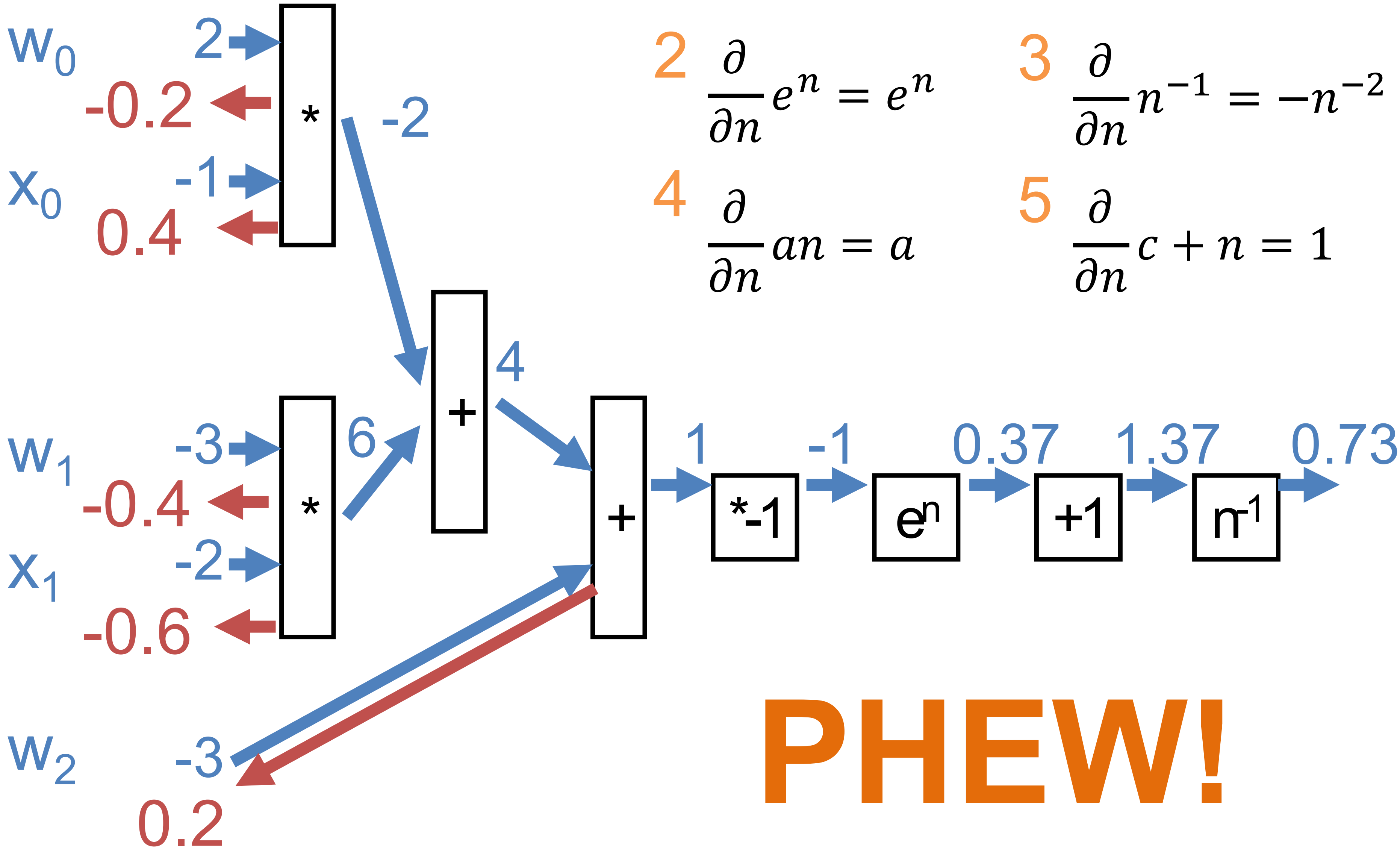
1 $\dfrac{\partial}{\partial n} mn = m$

2 $\dfrac{\partial}{\partial n} e^n = e^n$

3 $\dfrac{\partial}{\partial n} n^{-1} = -n^{-2}$

4 $\dfrac{\partial}{\partial n} an = a$

5 $\dfrac{\partial}{\partial n} c + n = 1$

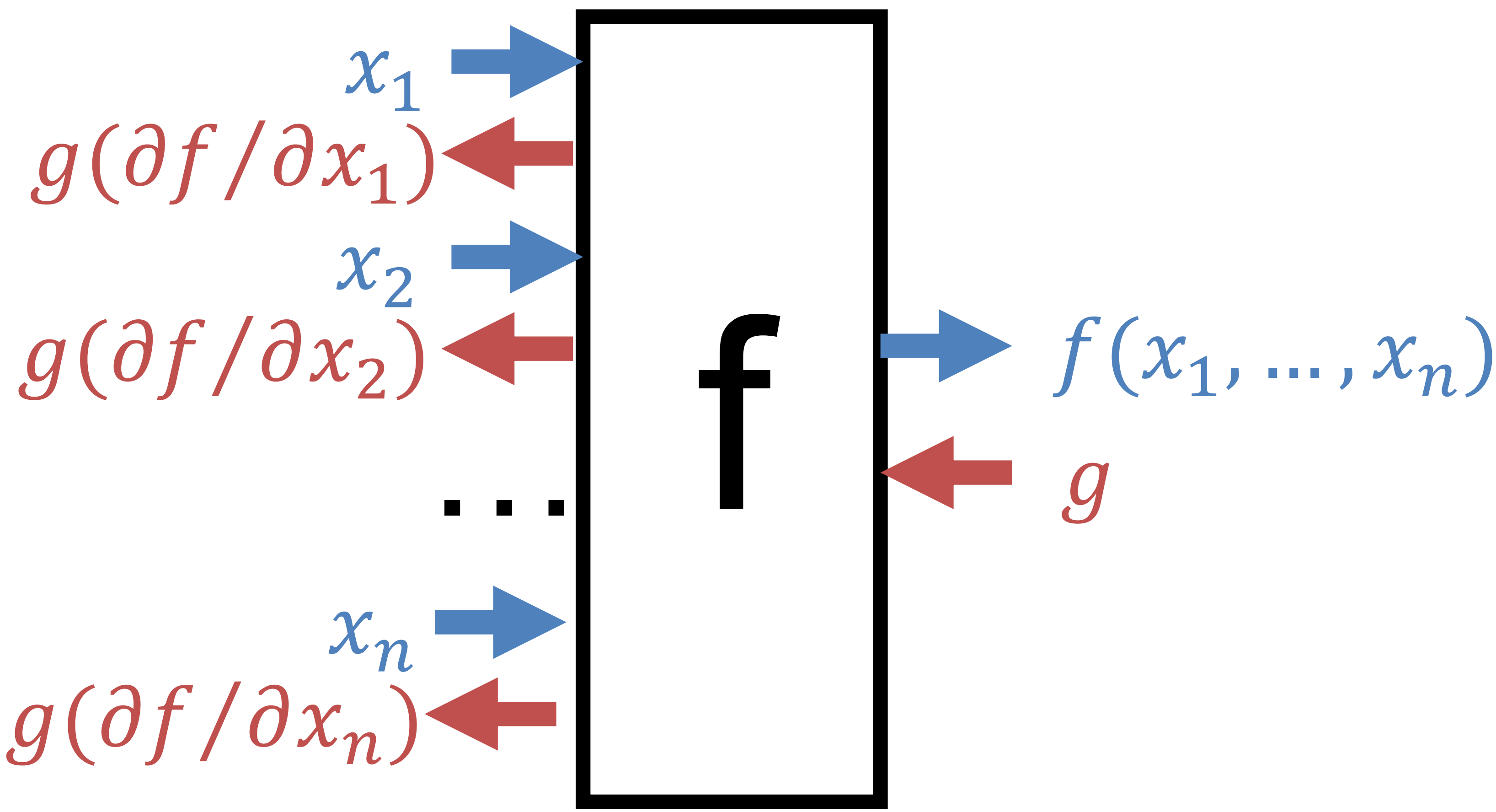


PHEW!

Example Credit: Karpathy and Fei-Fei

Summary

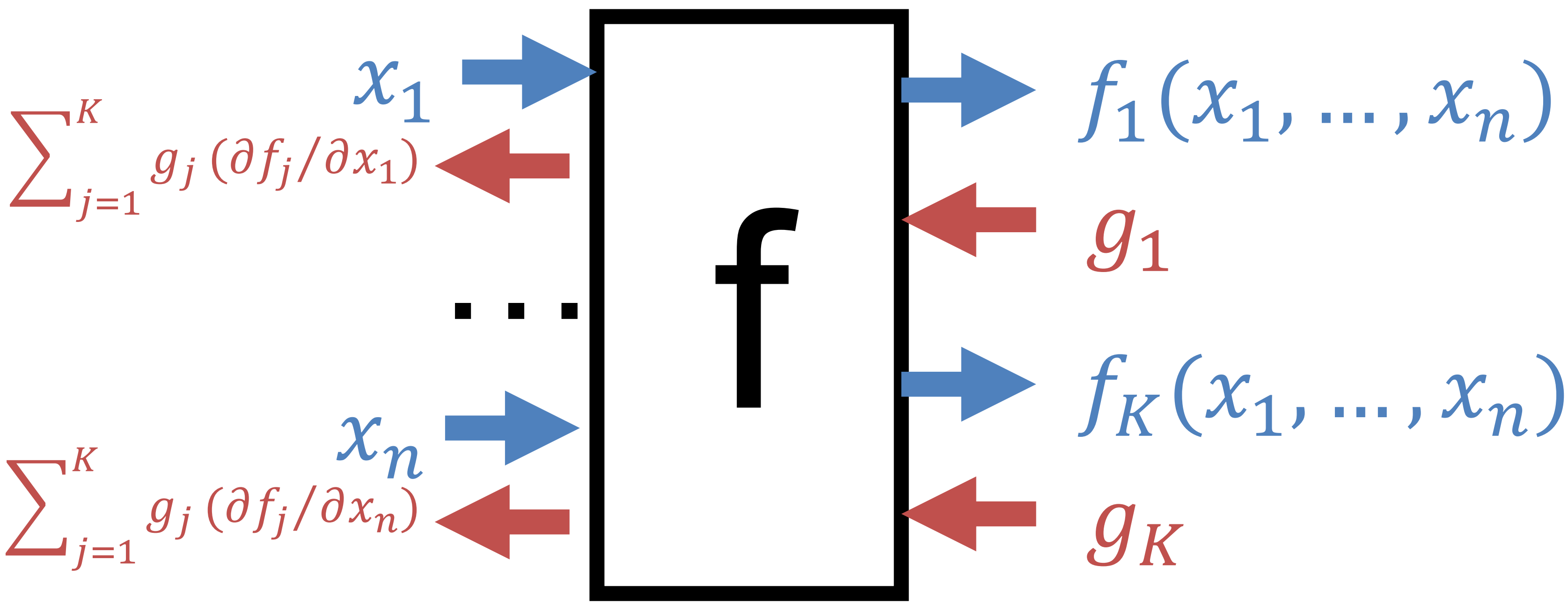
Each block computes backwards (g) * local gradient (df/dx$_i$) at the evaluation point



Multiple Outputs Flowing Back

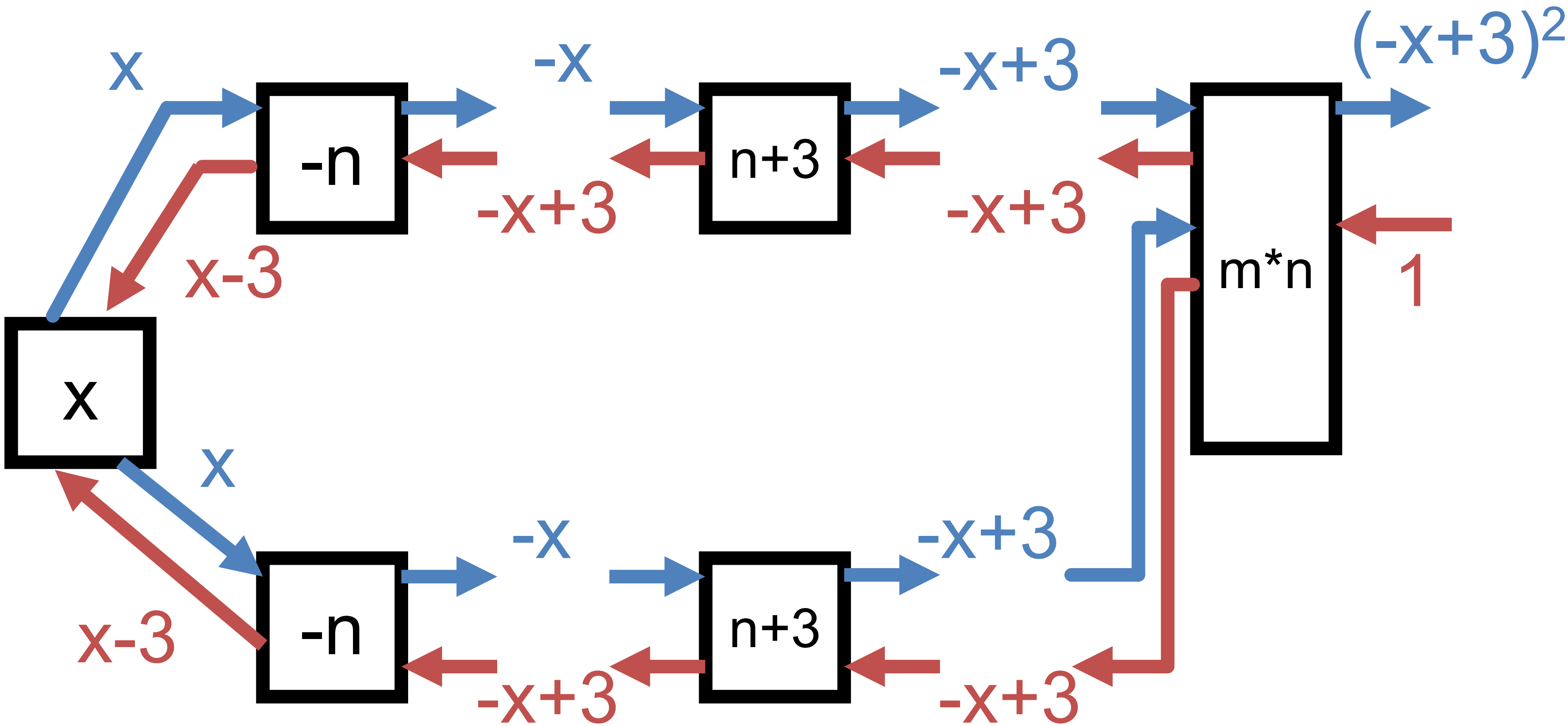
Gradients from different backwards sum up

$$\sum_{j=1}^{K} g_j \left(\partial f_j / \partial x_i\right)$$



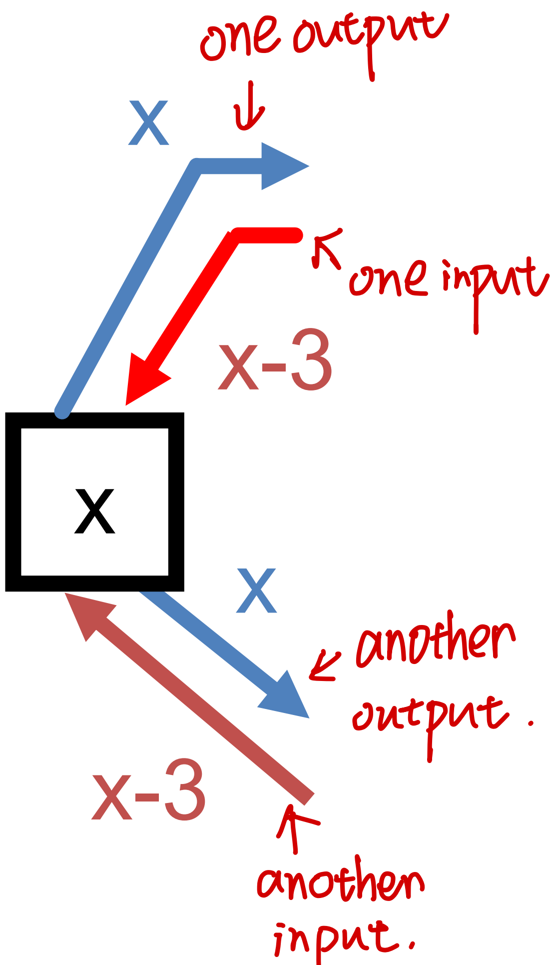
Multiple Outputs Flowing Back

$$f(x) = (-x+3)^2$$



Multiple Outputs Flowing Back

$$f(x) = (-x+3)^2$$

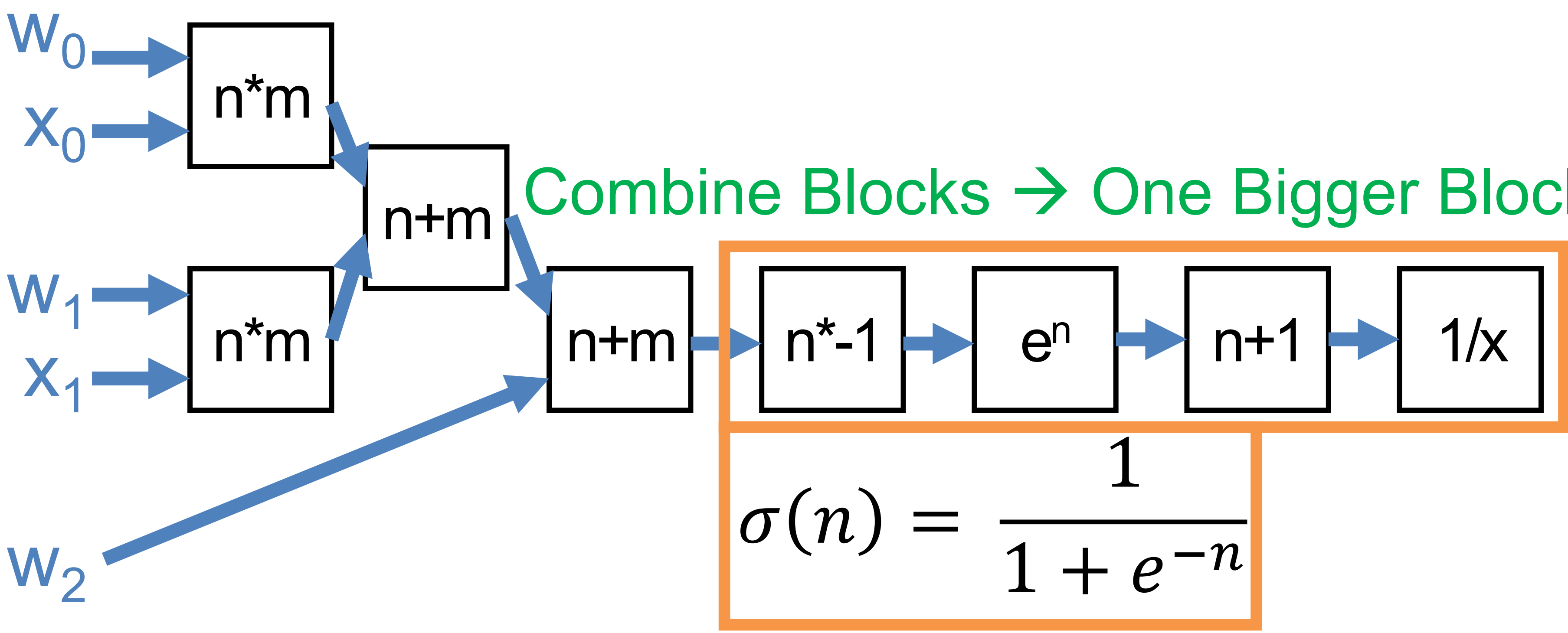


$$\frac{\partial f}{\partial x} = (x-3) + (x-3)$$

$$= 2x - 6$$

Does It Have To Be So Painful?

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$\sigma(n) = \frac{1}{1 + e^{-n}}$$

Example Credit: Karpathy and Fei-Fei

Does It Have To Be So Painful?

$$\sigma(n) = \frac{1}{1+e^{-n}}$$

$$\frac{\partial}{\partial n}\sigma(n) = \frac{e^{-n}}{(1+e^{-n})^2} = \left(\frac{1+e^{-n}-1}{1+e^{-n}}\right)\left(\frac{1}{1+e^{-n}}\right)$$

$$\frac{1+e^{-n}}{1+e^{-n}} - \frac{1}{1+e^{-n}} = 1-\sigma(n) \qquad \sigma(n)$$

$$= \big(1-\sigma(n)\big)\sigma(n)$$

For the curious

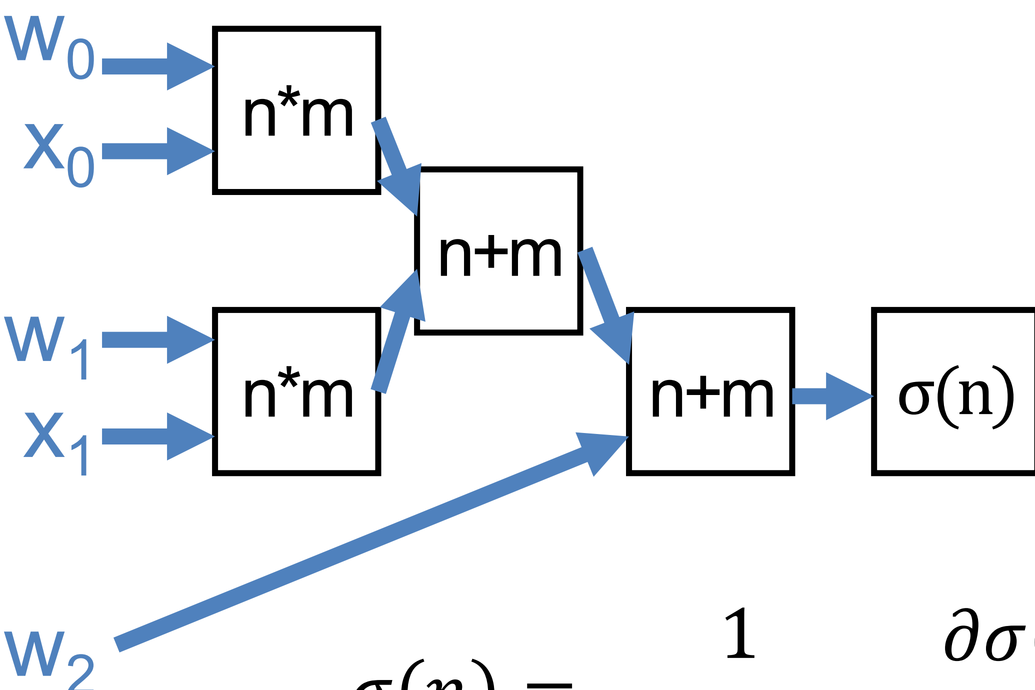
Line 1 to 2: $\frac{\partial}{\partial n}\sigma(n) = \left(\frac{-1}{(1+e^{-n})^2}\right) * 1 * e^{-n} * -1$

Chain rule: d/dx (1/x)*d/dx (1+x)* d/dx (e*x)*d/dx (-x)

Example Credit: Karpathy and Fei-Fei

Does It Have To Be So Painful?

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$\sigma(n) = \frac{1}{1 + e^{-n}} \quad \frac{\partial \sigma(n)}{\partial n} = (1 - \sigma(n))\sigma(n)$$

Example Credit: Karpathy and Fei-Fei

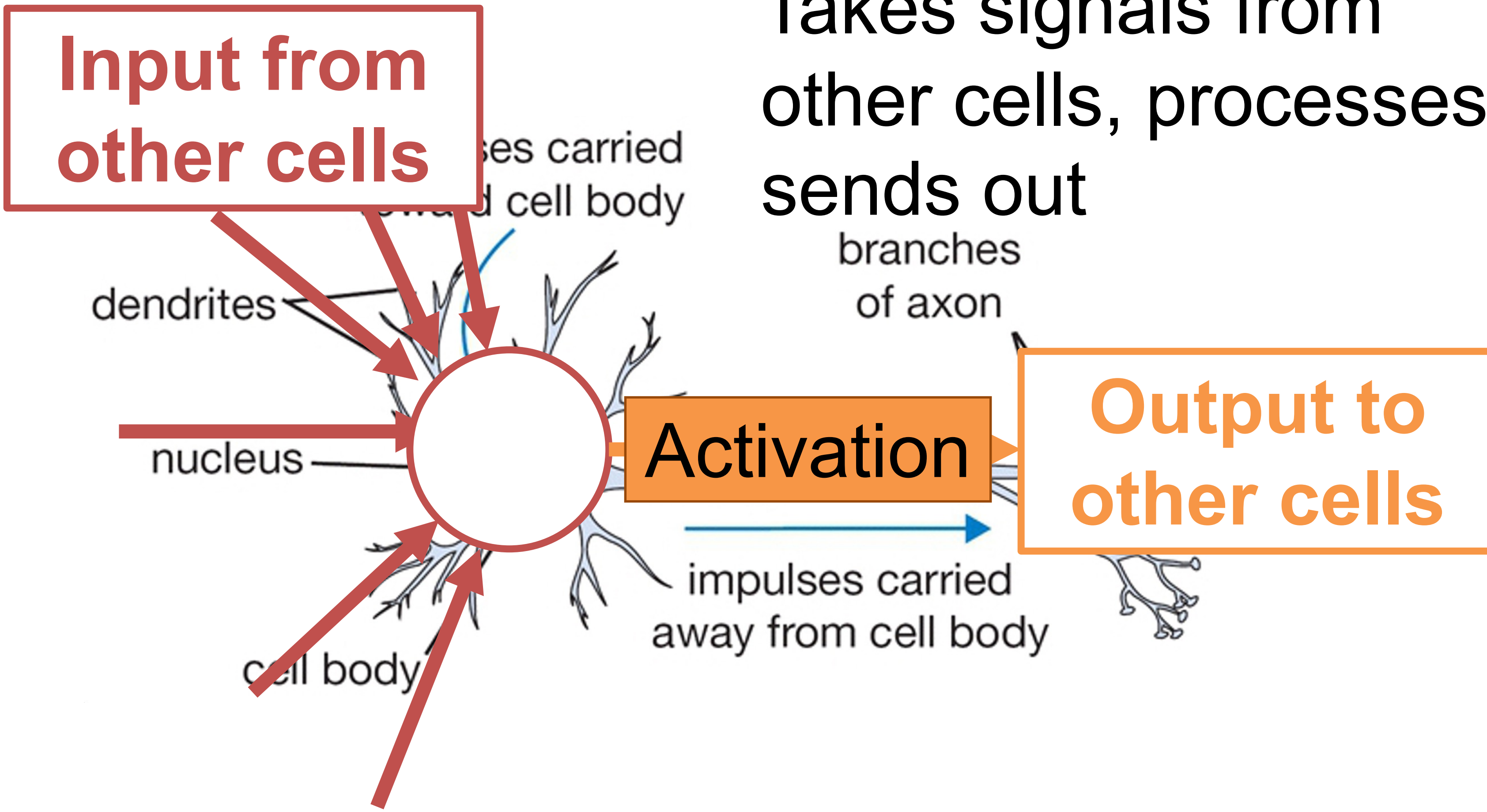
Does It Have To Be So Painful?

- Can pre-compute/hardcode the backward function.
- Reduce multiple blocks to one.
- Pick your functions carefully: existing code (e.g., Pytorch) is usually structured into sensible blocks

Neural Networks

Building Blocks

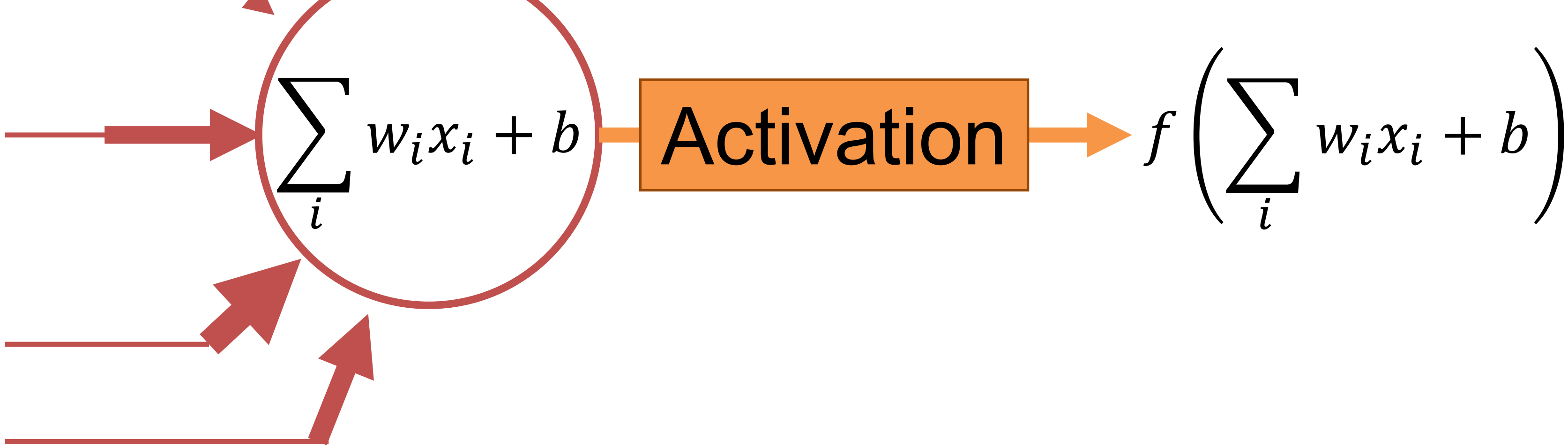
Takes signals from other cells, processes, sends out



Neuron diagram credit: Karpathy and Fei-Fei

Artificial Neuron

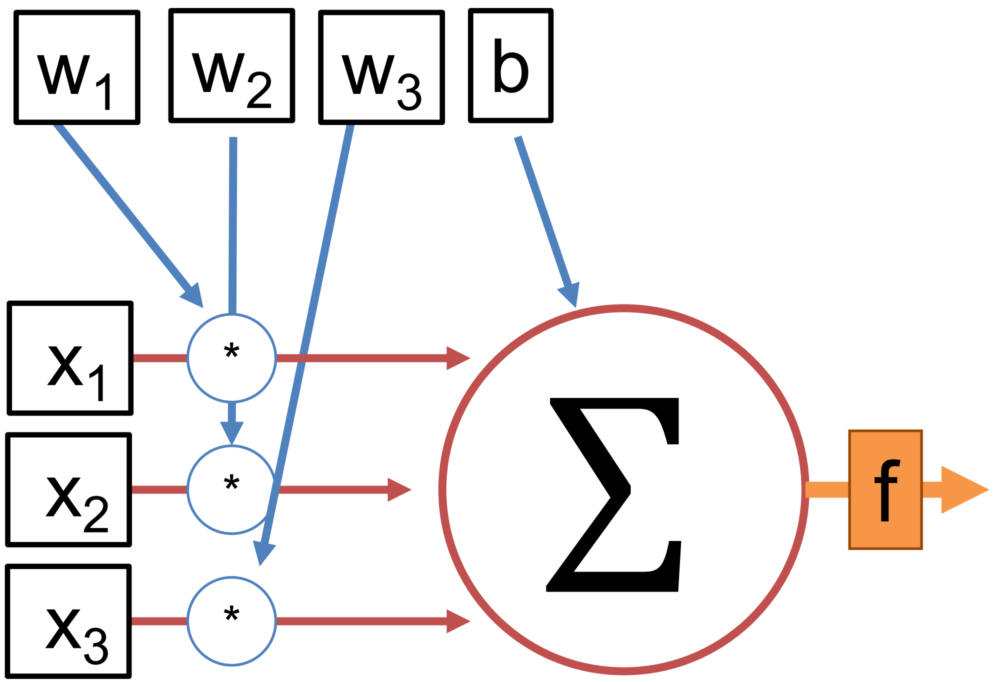
Weighted average of other neuron outputs passed through an activation function



Artificial Neuron

Can differentiate whole thing e.g., dNeuron/dx_1.

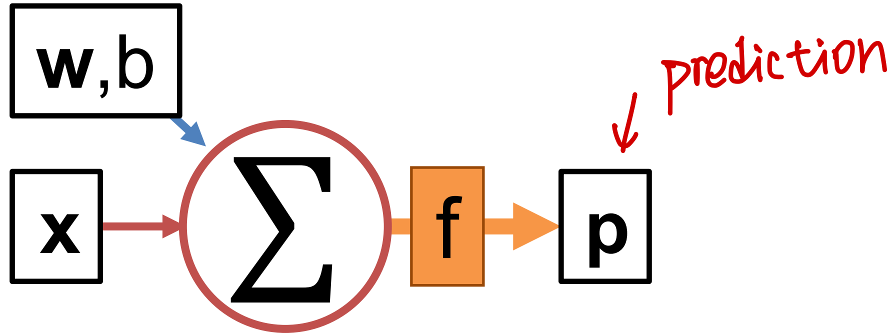
What can we now do?



Artificial Neuron

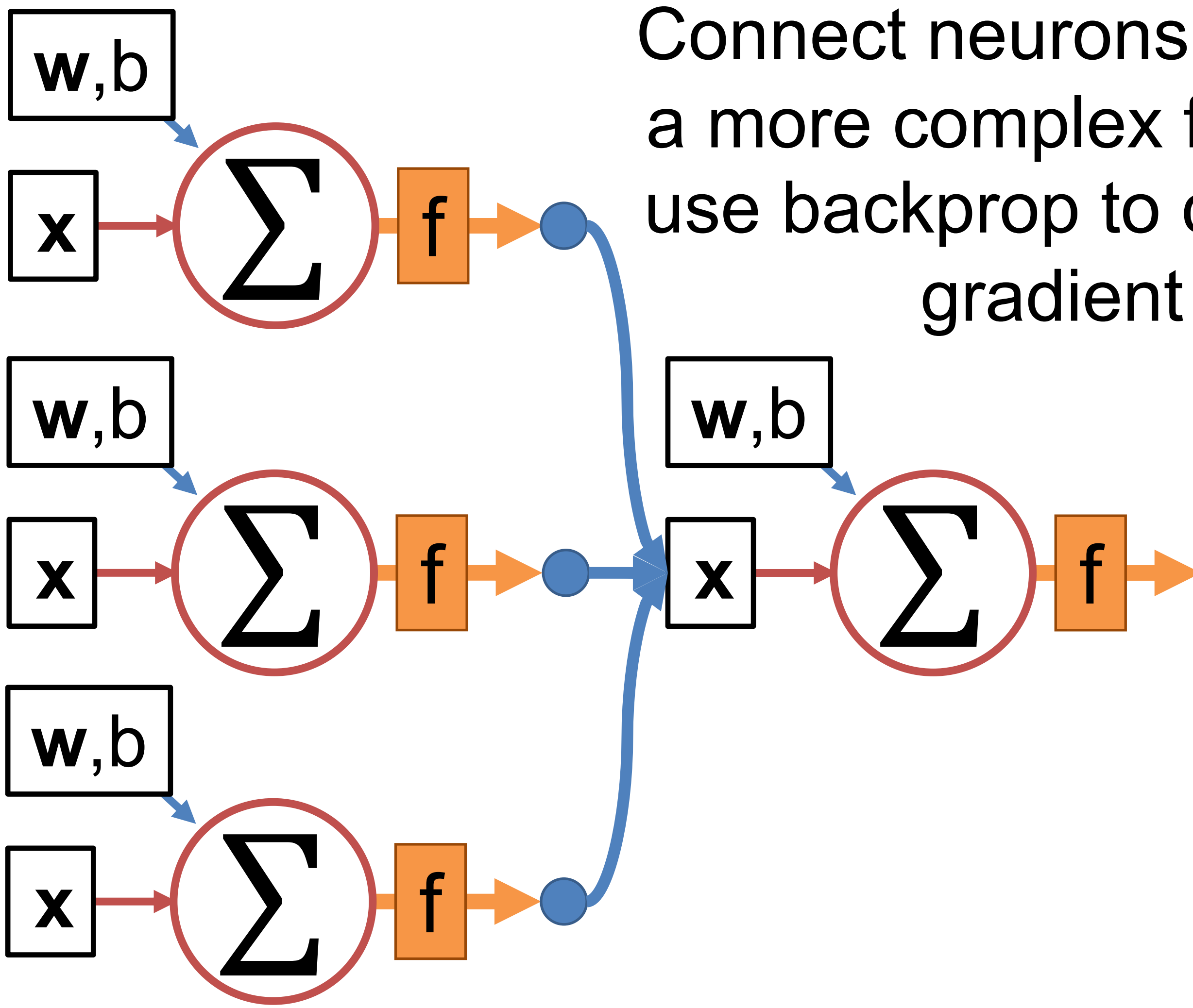
Each artificial neuron is a linear model + an **activation function** f

Can find **w**, b that minimizes a loss function with gradient descent

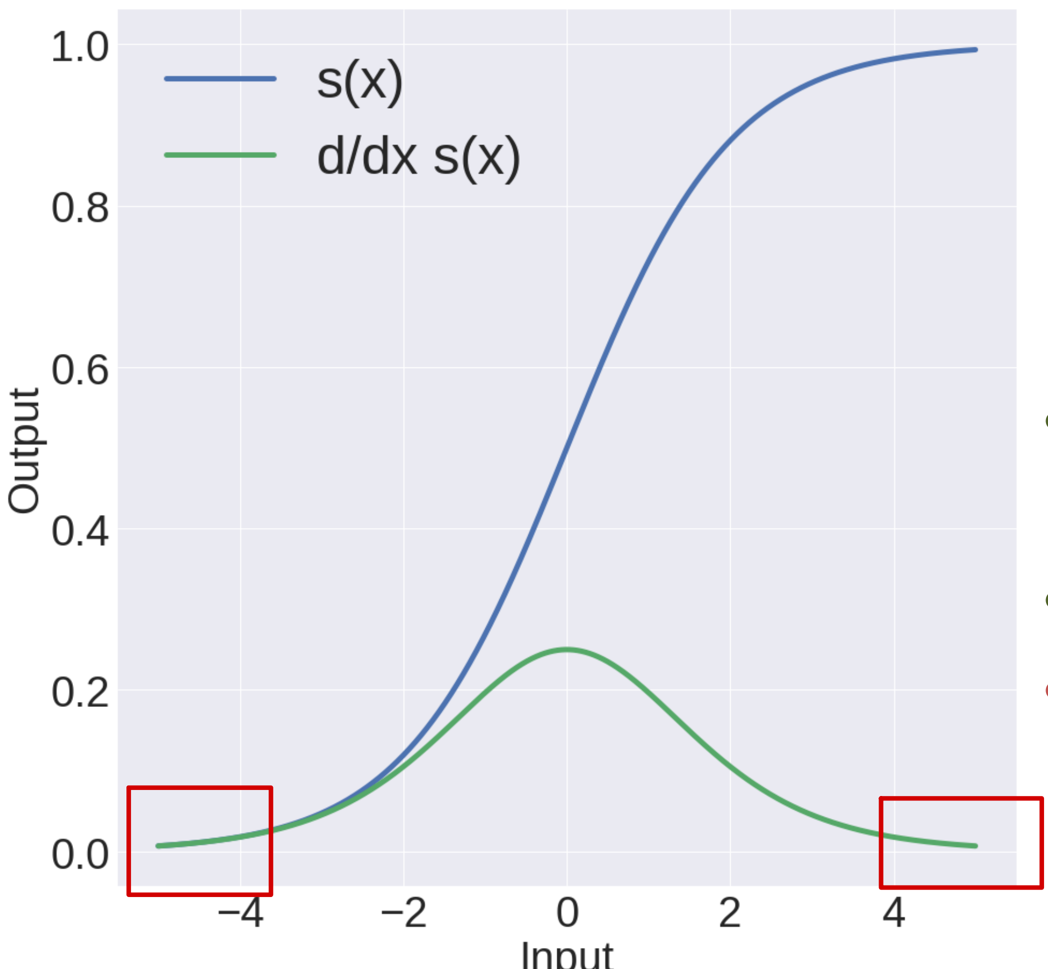


Artificial Neurons

Connect neurons to make a more complex function; use backprop to compute gradient



What's The Activation Function



1

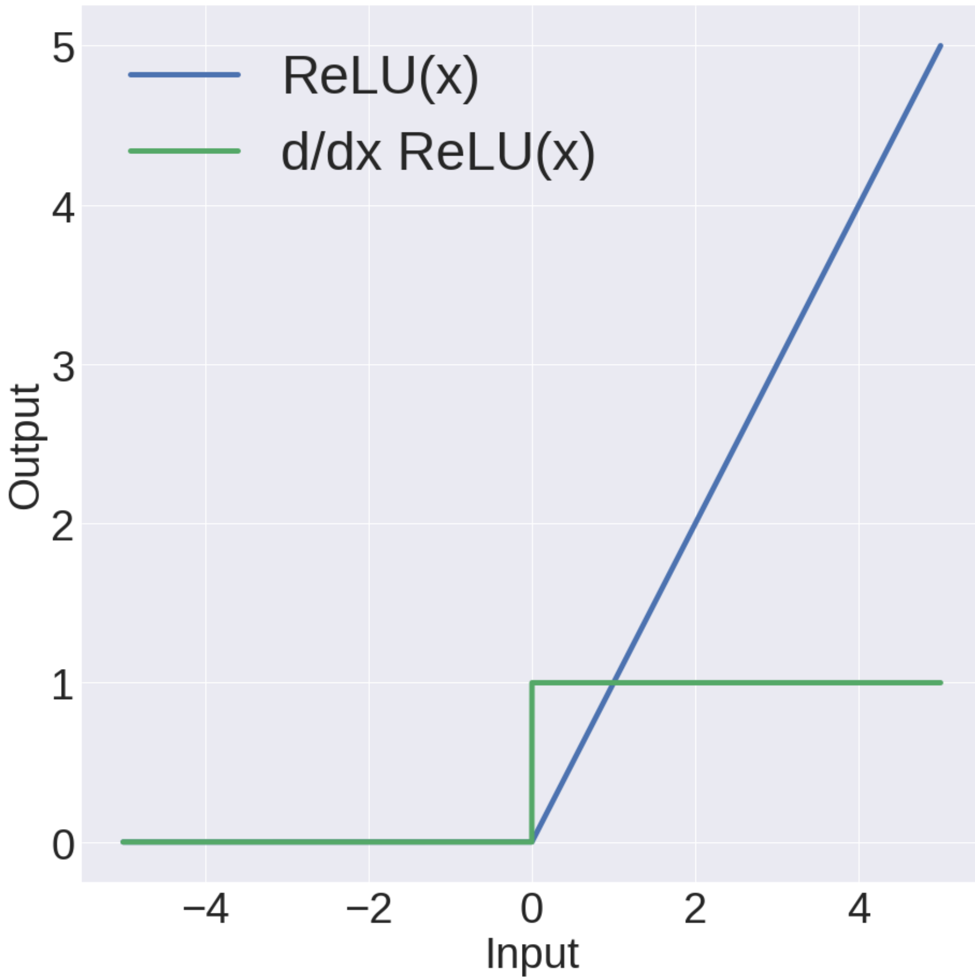
Sigmoid

$$s(x) = \frac{1}{1 + e^{-x}}$$

- Squashes things to (0,1)
- Nice interpretation
- Gradients are near zero if neuron is high/low

When the input is high or low, the gradients of this value is almost zero, so you not changing the variable during the training. And this neuro is basically dead.

What's The Activation Function



②

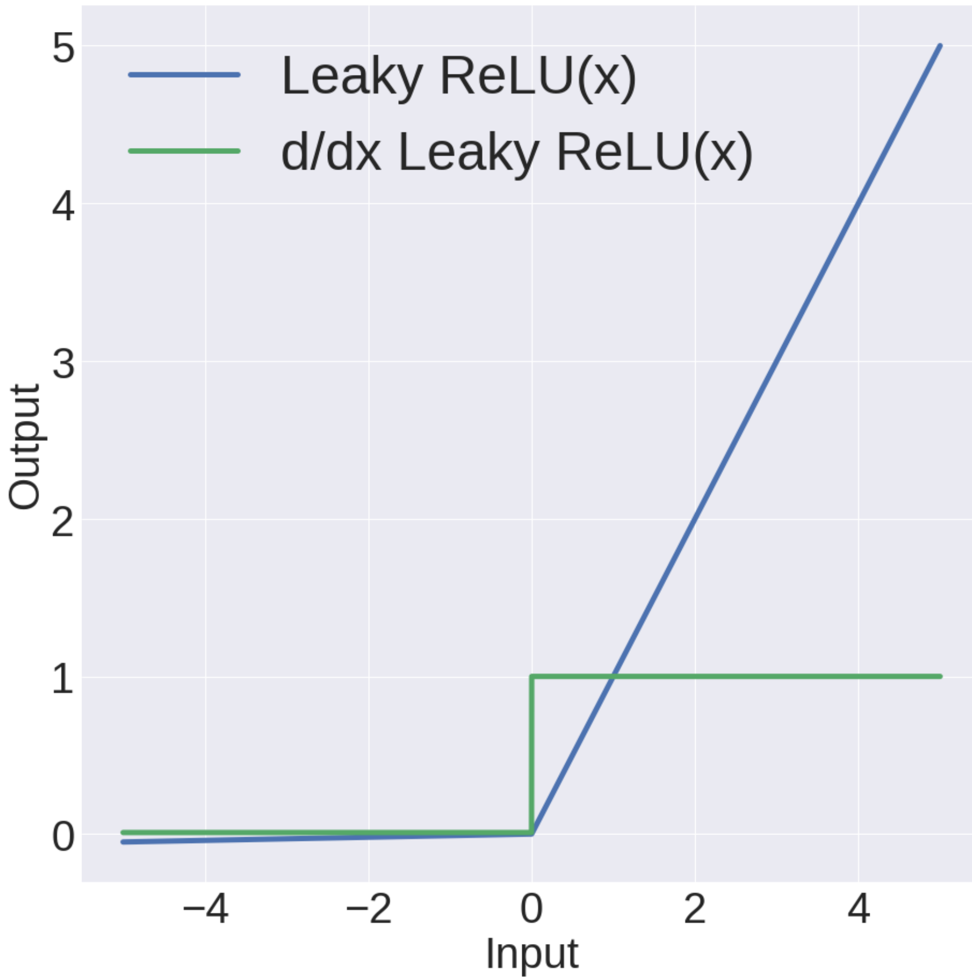
ReLU

(**<u>Re</u>**ctifying **<u>L</u>**inear **<u>U</u>**nit)

$$\max(0, x)$$

- Constant gradient
- Converges ~6x faster
- If neuron negative, zero gradient. Be careful!

What’s The Activation Function



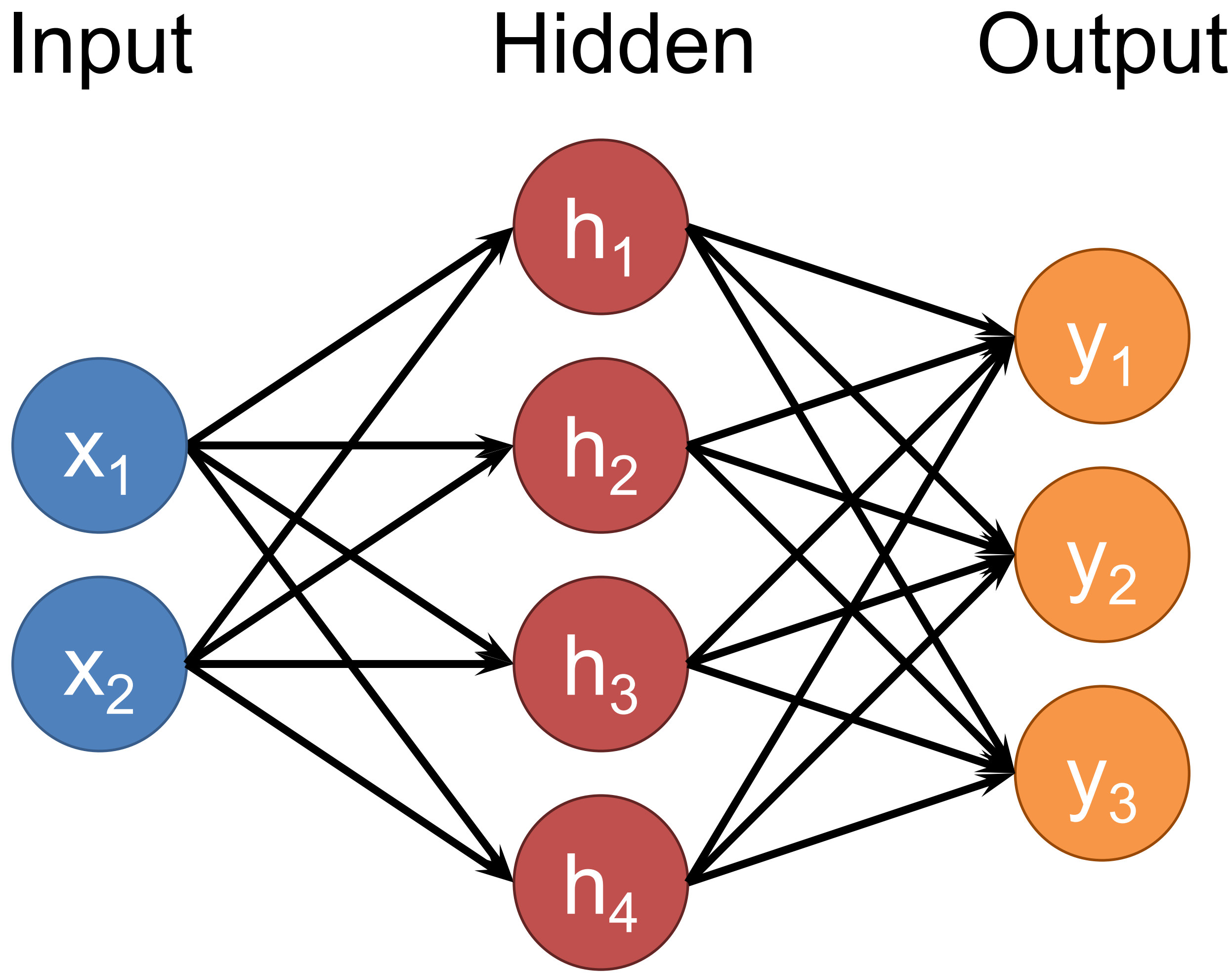
Leaky ReLU

(**Re**ctifying **L**inear **U**nit)

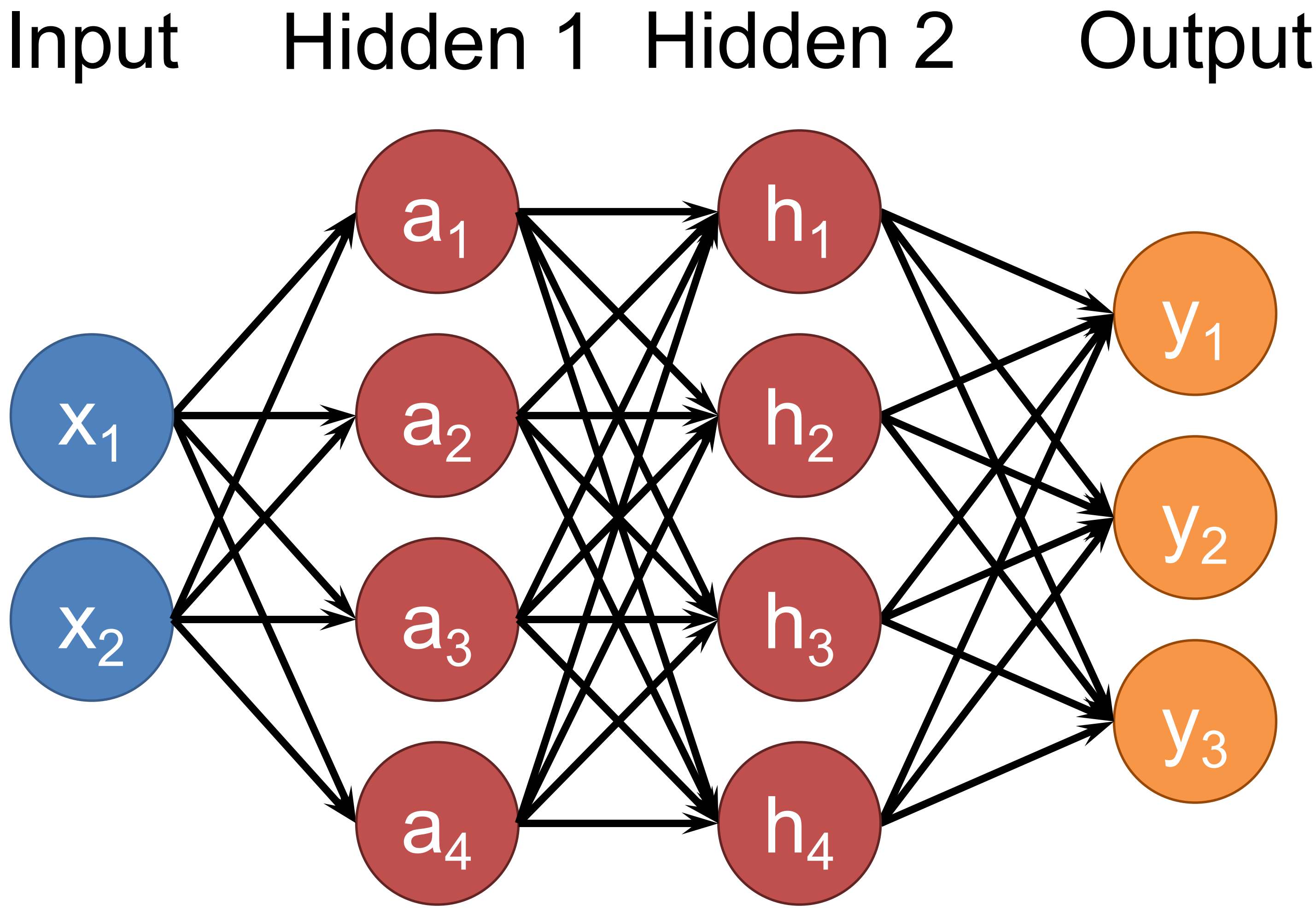
$$\begin{matrix} x : x \geq 0 \\ 0.01x : x < 0 \end{matrix}$$

- ReLU, but allows some small gradient for negative vales

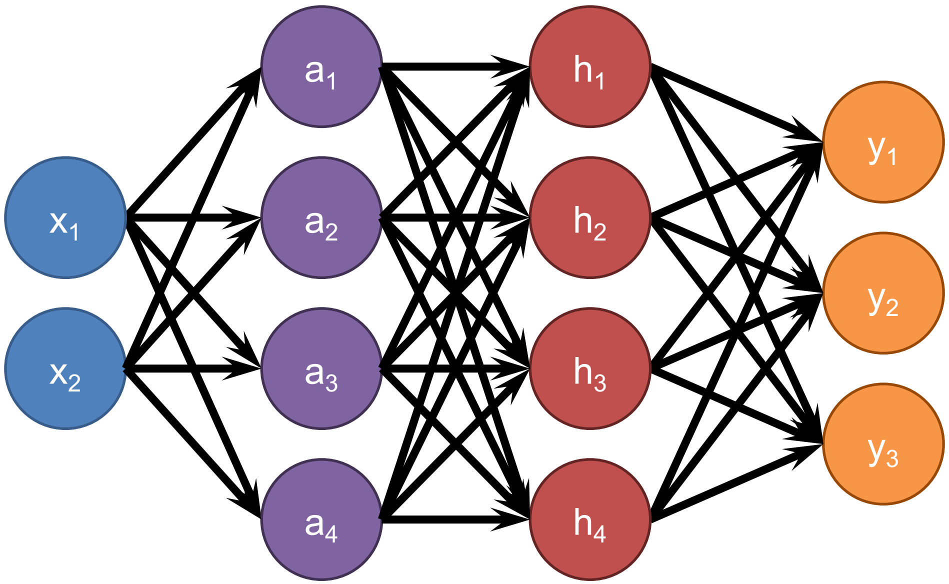
Setting Up A Neural Net



Setting Up A Neural Net

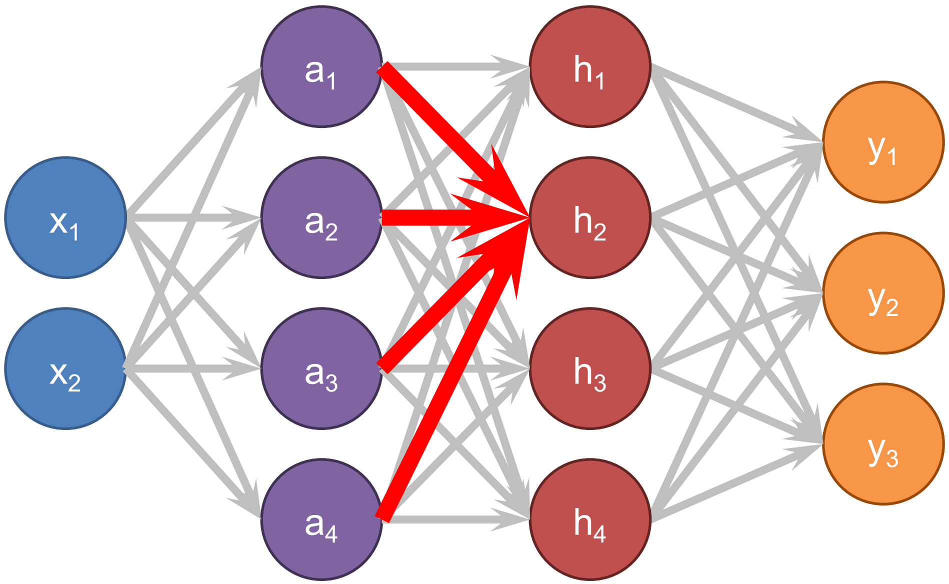


Fully Connected Network



Each neuron connects to each neuron in the previous layer

Fully Connected Network



$\boldsymbol{a}$ All layer *a* values

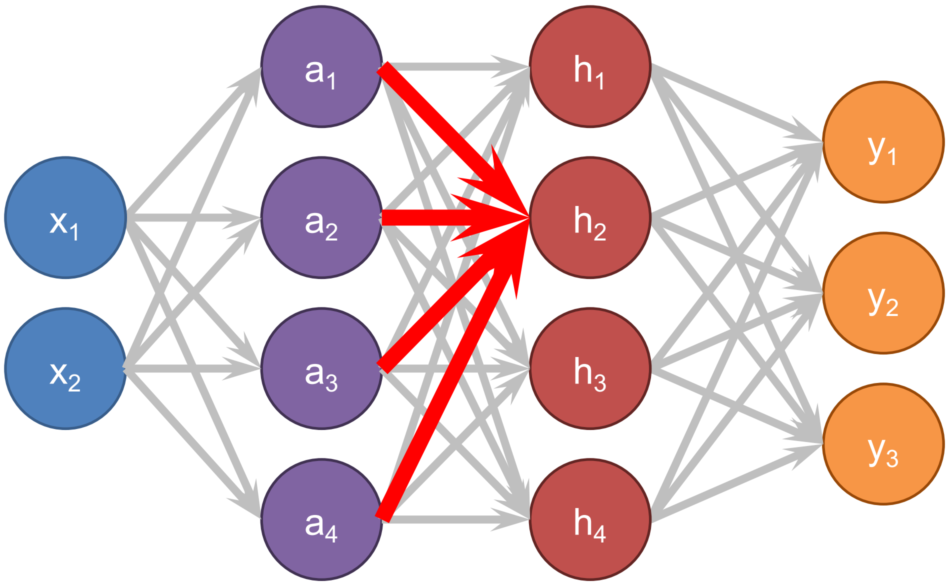
$\boldsymbol{w_i}, b_i$ Neuron i weights, bias

f Activation function

$$h_i = f(\boldsymbol{w_i^T}\boldsymbol{a} + b_i)$$

How do we do all the neurons all at once?

Fully Connected Network



$\boldsymbol{a}$ All layer *a* values

$\boldsymbol{w_i}, b_i$ Neuron i weights, bias

f Activation function

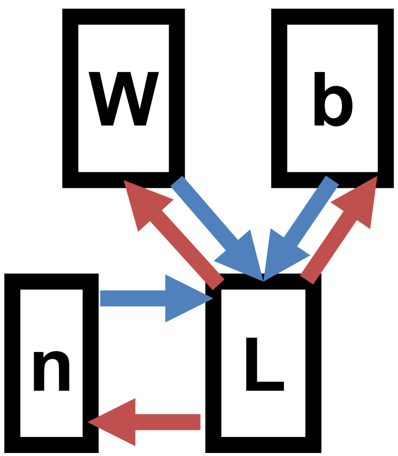
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{a} + \boldsymbol{b})$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = f\left(\begin{bmatrix} \longleftarrow & w_1^T & \longrightarrow \\ \longleftarrow & w_2^T & \longrightarrow \\ \longleftarrow & w_3^T & \longrightarrow \\ \longleftarrow & w_4^T & \longrightarrow \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \right)$$

Fully Connected Network

Define New Block: "Linear Layer"

(Ok technically it's Affine)

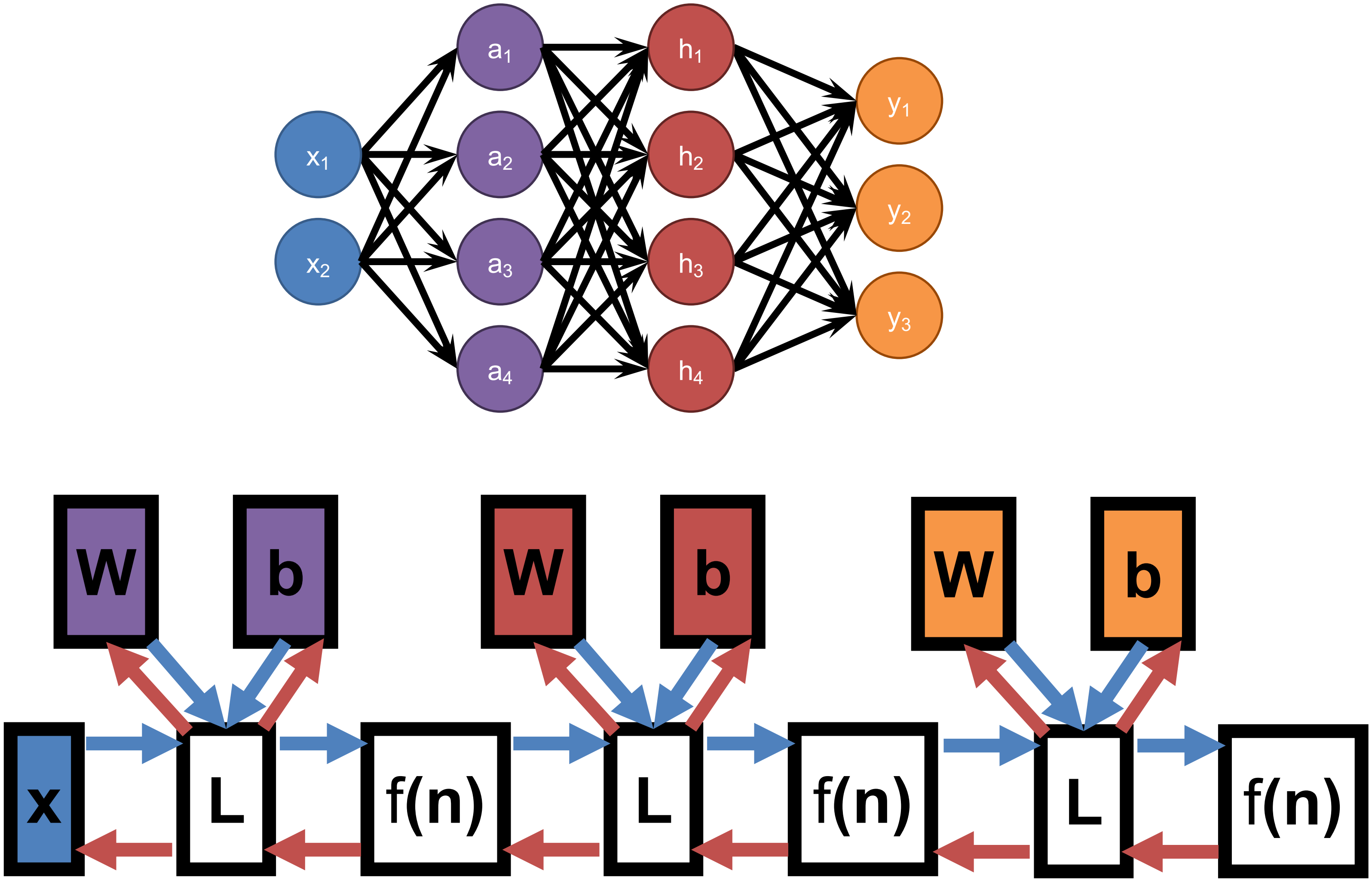


$$L(\boldsymbol{n}) = \boldsymbol{W}\boldsymbol{n} + \boldsymbol{b}$$

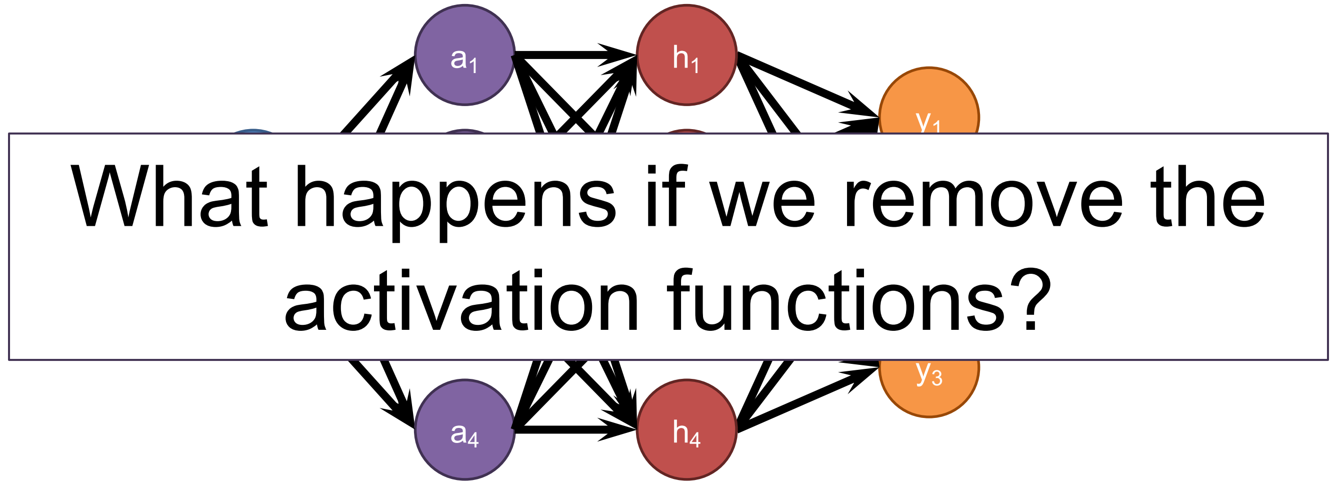
Can get gradient with respect to all the inputs

(do on your own; have to be able to do matrix multiply)

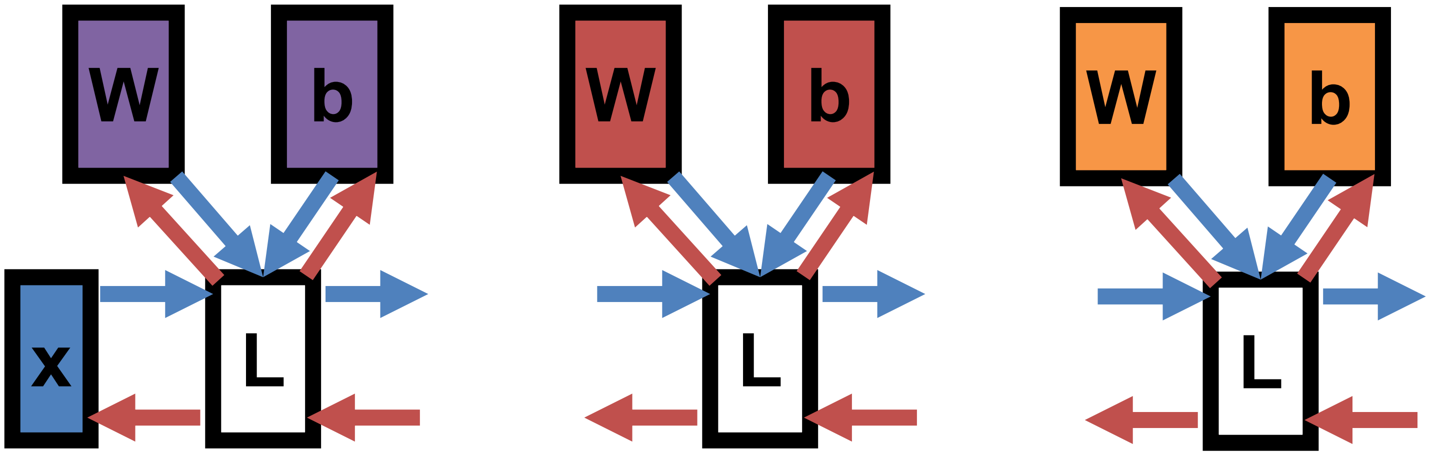
Fully Connected Network



Fully Connected Network



What happens if we remove the activation functions?



It collapses into one linear layer!

Neural Network

$$L(\boldsymbol{n}) = \boldsymbol{W}\boldsymbol{n} + \boldsymbol{b}$$

Something we understand very well.

$$L(\boldsymbol{n}) = \boldsymbol{ReLU}(\boldsymbol{W}\boldsymbol{n} + \boldsymbol{b})$$

Mysterious. Potentially conscious

Idea Credit: Jon Barron

Video Credit: Sora

Next Class:
Convolutional Neural Networks